# تعلم
## بدون تعقيد

# Visual Basic .net

- لفصائل والأهداف Classes and Objects
- دف البيانات ActiveX Data Object(ADO.NET)
- دد الأشكال والتجريد Polymorphism and Abstraction
- فحات الخادم النشطة Active Server Pages(ASP.NET)
- العديد ...


مهندس
مصطفى ماجد

مهندس
قدرى حسين




تعلم بدون تعقيد ..

# Visual Basic .net




المركز الرئيسي : 11 شارع د/محمد رافت – محطة الرمل – الإسكندرية
تليفون وفاكس : 4838326 (03)(+2)
موبايل : 0101634294 (+2) – 0123357844 (+2)
Email: info@egyptbooks.net
URL: www.egyptbooks.net

إعداد وتأليف

م/مصطفى ماجد م/قدري حسين



وقل اعملوا فسيرى الله عملكم ورسوله والمؤمنون

صدق الله العظيم




**طبعة يناير 2005**

**رقم الإيداع**

**2005/1912**

**ISBN**

**977-17-1971-8**

بسم الله الرحمن الرحيم

## مقدمة

أراد شخص من الأشخاص شراء كتاب يقوم بتغطية لغة Visual Basic وتفقد المكتبات بحثاً عن كتاب مناسب ، فوجد نوعيتين من الكتب: النوعية الأولي ذات حجم ضخم تحتاج إلي عضلات قوية لمجرد حملها كما أن سعرها عالي جداً ، والنوعية الثانية تعتبر روايات للجيب ولا يمكن بكل المقاييس أن تحتوي علي المعلومات الكافية لتعلم لغة قوية مثل لغة Visual Basic ، فاحتار هذا الشخص في إيجاد الكتاب المناسب ، فالكتاب الضخم لا يناسبه لأنه سيحتوي علي كمية ضخمة من المعلومات المتخصصة التي لا يحتاجها هذا الشخص كما أنه لا يوجد لديه الوقت الكافي لقراءة الكتاب بالكامل بالإضافة إلي أن سعر الكتاب يكفي لخلق شهور من التقشف لدفع ثمنه ، كما أن كتب الجيب في مجال البرمجة لا يمكن أن تقوم بتغطية المعلومات الأساسية لتعلم اللغة.

ومن هنا جاءت فكرة إعداد هذه السلسلة المبسطة والتي تقدم للقارئ المعلومات الأساسية التي يحتاجها دون حشو ودون الدخول في تفاصيل تحتاج إلي متخصصين أصحاب خبرة كبيرة وتؤدي غالباً إلي تعقيد القارئ من اللغة وبالتالي تضيع عليه فرصة كبيرة للاستفادة من إمكانات اللغة ، كما تم تصميم حجم الكتاب بحيث يكون متوسطاً ومتوازناً بين الحجم الصغير والكبير وبالتالي يكفي لبناء قاعدة من المعلومات تؤهل أي قارئ لتعلم واستخدام اللغة بشكل جيد ، كما يستطيع القارئ أن ينهي الكتاب في أسرع وقت ممكن.

ويعد مجال البرمجة من أهم التطبيقات في الواقع العملي وذلك لما له من أهمية قصوي في تصميم البرامج التي تنفذ العديد من المهام بسرعة ودقة.

وتعد لغة Visual Basic من أهم لغات البرمجة الموجودة حالياً وذلك لما تتميز به هذه اللغة من سهولة وبساطة في الاستخدام بدون أي تعقيد ، كما يوجد جزء كبير من لغة Visual Basic مخصص لتطوير المواقع علي شبكة الإنترنت Internet ، ولا ننسي الدور الرائد الذي تقوم به شركة مايكروسوفت Microsoft في دعم هذه اللغة حتي أصبحت لغة Visual Basic هي اللغة المفضلة للعديد من المبرمجين في سوق العمل.

ويقوم هذا الكتاب بعرض إمكانات لغة Visual Basic في صورة سهلة ومبسطة وسريعة بحيث تقدم للقارئ أكبر قدر ممكن من المعلومات في أبسط صورة وأقل وقت ممكن.

وتسهيلاً علي القارئ ، وكإضافة وخدمة جديدة نقدمها للقارئ العزيز ، فقد تم وضع جميع أمثلة هذا الكتاب علي شبكة الإنترنت Internet ويمكن للقارئ أن يقوم بتنزيلها Download مجاناً من موقع المؤلفين التالي:

**http://www.geocities.com/mostafadarsh7/downloads.html**

وبالتالي يتم توفير وقت كبير علي المستخدم في تطبيق أمثلة هذا الكتاب.

وقد تم تصميم محتويات هذا الكتاب بحيث تنقسم إلي أربعة أجزاء:

- الجزء الأول (الفصول من الأول إلي الرابع) يقوم بتغطية الأساسيات العامة للغة.
- الجزء الثاني (الفصول من الخامس إلي الثامن) يقوم بتغطية مبادئ البرمجة موجهة الهدف Object Oriented Programming والاستثناءات Exceptions.
- الجزء الثالث (الفصول من التاسع إلي العاشر) يقوم بشرح كيفية تصميم النماذج Forms وكيفية التعامل مع الأحداث Event لتصميم واجهة متكاملة للبرنامج.

- الجزء الرابع (الفصول من الحادي عشر حتي الثاني عشر) يقوم بشرح كيفية برمجة قواعد البيانات Databases سواء علي الأجهزة الشخصية أو علي شبكة الإنترنت Internet من خـــلال تقنيـــة هـــدف البيانـــات ActiveX Data Object (ADO . NET) وصفحات الخادم النشطة Active Server Pages (ASP .NET).

وأخيراً نتمني من الله أن يجد القارئ في هذا الكتاب ما نرجوه له ومـــا يرجوه لنفسه.

**ونسأل الله العون والتوفيق لاستكمال هذه السلسلة ،،،**

م/ مصطفي ماجد محمود

م/ قدري طلعت حسين

يوليو 2004

# اتصل بالمؤلفين
# Contact the Authors

يمكنك الاتصال بالمهندس/ مصطفي ماجد محمود عن طريق الدخول إلـــي موقعه الشخصي علي شبكة الإنترنت Internet:

http://www.geocities.com/mostafadarsh7
http://www.geocities.com/mostafamaged

كما يمكنك مراسلته بالبريد الإليكتروني E-mail التالي:

mostafadarsh7@yahoo.com
mostafamaged1978@hotmail.com

كما يمكن مراسلة المهندس / قدري طلعت حسين بالبريد الإليكترونـــي E-mail التالي:

eng_kadry@hotmail.com

تم وضع جميع أمثلة هذا الكتاب علي شبكة الإنترنت Internet ويمكن للقارئ أن يقوم بتنزيلها Download مجانا من الموقع :.

**http://www.geocities.com/mostafadarsh7/downloads.html**

أو

**www.egyptbooks.net**

والمؤلفان يرجوان كل قارئ أن يقدم مقترحاته بكل حرية وبدون الشـــعور بـــأي حرج.

# إهداء

- إلي أبي وأمي وإخوتي ، إلي هذه العائلة العظيمة التـــي وفــرت لـــي كــل الإمكانات التي ساعدتني في الوصول إلي هذه المكانة التي ســاعدتني فـــي إخراج هذا الكتاب.
- إلي كل من يريد الدخول في عالم البرمجة.
- إلي كل من يريد تعلم لغة Visual Basic.
- إلي كل من ساعدني في إخراج هذا الكتاب.
- أخيراً أتمني من الله أن يكون قد وفقنا في إخراج هــذا العمــل وأن تكــون الاستفادة أقصي ما يمكن لقارئ هذا الكتاب.

م/ مصطفي ماجد محمود

---

- حياة الانسان عبارة عن محطات زمنية متصلة ومترابطة. وفى كل محطـــة يقابل ما قدر الله له أن يراهم ويتعامل معهم. وفى كل محطة وجدت أنـــاس أعانونى وعلمونى وتأثرت بهم ، ولكل هؤلاء الناس أهدى جهدى المتواضع لعله يكون يوماً كلمة شكر لم أقلها.
- كما أهديه لكل قارئ يطلب العلم وأدعو الله أن يكون هذا الكتـــاب خطـــوة تساعده على الوصول إلى ما يعلى به شأن نفسه وأمته وأدعو الله ان يكـــون فى ميزان حسناتنا إن شاء الله.

م/ قـــدرى طلعت حسين

# الفصل الأول

## مقدمة إلي فيجوال بيسك
## Introduction to Visual Basic

في هذا الفصل نتعرف علي أساسيات برنامج Visual Studio .NET ونتعرف علي المفاهيم الأساسية في البرنامج ، كما نتعرف علي كيفية إنشاء مشروع Project بسيط يقوم بطباعة رسالة علي شاشة الدوس Dos كما نتعرف علي واجهة البرنامج وأهم مكوناتها وكيفية التعامل معها وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. لغة فيجوال بيسك Visual Basic.
3. خصائص ومميزات Visual Studio ..NET
4. إطار عمل دوت نت NET Framework..
5. لغة وقت التنفيذ العامة Common Language Runtime (CLR).
6. تشغيل البرنامج.
7. مكونات واجهة البرنامج Components of the IDE Window.
8. تنفيذ المشروع Project.
9. إغلاق المشروع Clos،ng Solution.
10. إنهاء البرنامج Exit.
11. فتح المشروع Opening Solution.
12. ملخص الفصل.

مقدمة إلي فيجوال بيسك
Introduction to Visual Basic

## مقدمة:

- يعتبر مجال البرمجة من أهم وأقوي التطبيقات في الواقع العملي وذلك لما له من أهمية قصوي في تصميم البرامج التي تنفذ العديد من المهام بسرعة ودقة.
- ويمكن تعريف البرنامج بأنه مجموعة من الأوامر المكتوبة بشكل معين ، وهذه المجموعة من الأوامر تسمي بلغة البرمجة ، وتعتبر مجموعة الأوامر الموجودة في هذه اللغة أشبه بمجموعة الكلمات الموجودة في أي قاموس لغوي حيث أن لكل أمر معني وغرض معين.
- وكان من الطبيعي أن تتعدد لغات البرمجة بشكل كبير جداً منذ نشأة علم البرمجة ، ولكن يمكن بشكل عام تقسيم لغات البرمجة إلي ثلاثة أقسام:
  - لغات الآلة Machine Languages.
  - لغات التجميع Assembly Languages.
  - لغات المستوي العالي High-level Languages.
- وتعتبر لغات الآلة Machine Languages هي اللغات التي تتعامل مباشرة بلغة الآلة التي نقوم بتشغيلها وتتكون من مجموعة من الأصفار والآحاد لتقوم بتنفيذ الأوامر بطريقة إليكترونية من خلال الدوائر الإليكترونية العديدة في الآلة.
- ولكن مع انتشار أجهزة الكمبيوتر ، بدأت مشاكل لغات الآلة Machine Languages في الظهور ، وتتمثل هذه المشاكل في أن لغة الآلة تتعامل مع نوع واحد فقط من الآلات وبالتالي لا تصلح كلغة برمجة عامة لجميع المبرمجين ، هذا بالإضافة إلي صعوبتها وعدم القدرة علي فهم أوامرها من خلال قراءة الأوامر ، لأن أي انسان لن يستطيع فهم لغة لا يعرفها (تخيل أنك تقرأ الصينية أو اليابانية) وينطبق نفس هذا الكلام علي لغة الآلة.

- ومن هنا بدأت لغات التجميع Assembly Languages في الظهور عن طريق استخدام اختصارات لبعض مفردات اللغة الإنجليزية ، وبذلك أصبحت القدرة علي فهم الأوامر أكثر سهولة ، ولكن مشكلة هذا النوع من اللغات أنه يعتمد علي كتابة العديد من الأوامر لتنفيذ أي مهمة مهما كانت بساطتها.
- ومن هنا وصلنا إلي لغات المستوي العالي High-level Languages التي تعتمد علي استخدام مفردات اللغة الإنجليزية (وليس مجرد اختصارات) لكتابة الأوامر ، كما أصبح تنفيذ المهام يتم بشكل أكثر سهولة وبساطة.
- ومن أشهر لغات المستوي العالي High-level Languages تأتي لغة سي وسي++ C/C++ وفيجوال بيسك Visual Basic وجافا Java.
- ونتيجة للتطوير المستمر في لغات البرمجة بسبب ظهور مجالات وتقنيات جديدة بالإضافة للتقدم التكنولوجي وانتشار أجهزة الكمبيوتر ، فقد تم إصدار لغة فيجوال بيسك Visual Basic بشكل جديد لتواكب هذا التقدم.

## لغة فيجوال بيسك Visual Basic:

تعد لغة فيجوال بيسك Visual Basic من أبسط لغات البرمجة التي ظهرت بشكل جديد في الآونة الأخيرة ، وهي أحد اللغات ضمن مجموعة فيجوال ستوديو Visual Studio والذي تم إصدار أحدث نسخة منه باسم Visual Studio ..NET

وقد بدأ الإعلان عن Visual Studio .NET في يونيو 2000 ويعد هو الإصدار السابع لهذه المجموعة من لغات البرمجة التي تحتوي أيضاً علي لغة سي شارب C# وفيجوال سي++ Visual C++.

## خصائص ومميزات Visual Studio .NET:

1. عدم الاعتماد علي نظام تشغيل معين Operating System Neutral:

قامت شركة مايكروسوفت Microsoft بتصميم تطبيقات Visual Studio .NET. بحيث لا يقوم التطبيق باستدعاء أي وظيفة من نظام التشغيل Operating System ، وتطمح مايكروسوفت Microsoft من هذه الخطوة إلي تشغيل تطبيقاتها علي أي نظام تشغيل Operating System ولا يكون التشغيل مقتصراً فقط علي نظام النوافذ Windows.

2. دعم عدد كبير من اللغات Wide Language Support:

وهذه الخاصية في غاية الأهمية ، فحيث أن لغة Visual Basic ليست هي اللغة الوحيدة الموجودة في مجال البرمجة ، إذن فمن الطبيعي أن المطورين في بعض الشركات قد قاموا باستخدام أي لغة أخري لتصميم التطبيقات ، ولذلك قامت شركة مايكروسوفت Microsoft بدعم عدد كبير من اللغات الأخري بحيث يمكن كتابة الأوامر بأي لغة برمجة واستخدامها في بناء التطبيقات في Visual Studio .NET حيث قامت شركة مايكروسوفت Microsoft بدعم أكثر من 20 لغة مثل Java و Pascal و Perl و JScript وغيرها.

3. دعم كامل لتطبيقات وخدمات الإنترنت Internet:

قامت شركة مايكروسوفت Microsoft بإنشاء مجموعة من الفصائل Classes المخصصة لبرمجة مواقع الإنترنت Internet والتعامل مع قواعد البيانات (من خلال تقنية صفحات الخادم النشطة Active Server Pages ASP) وبذلك يمكن استخدام العديد من التقنيات الحديثة من داخل Visual Studio .NET مثل استخدام XML و SOAP (تستخدم هذه التقنيات في بناء خدمات الوب Web Services).

4. دعم كامل لتطبيقات الهاتف المحمول:

قامت شركة مايكروسوفت Microsoft بتقديم ما يسمي بأدوات الإنترنت للهاتف المحمول Mobile Internet Tools (MIT) والتي يمكن بواسطتها تصميم تطبيقات كاملة للهاتف المحمول.

## إطار عمل دوت نت .NET Framework:

المقصود بإطار عمل دوت نت .NET Framework هو مجموعة الفصائل Classes التي صممتها شركة مايكروسوفت Microsoft لتساعد المبرمجين علي سرعة وسهولة كتابة وإنشاء التطبيقات ، ويمكن تقسيم هذه الفصائل Classes إلي أربعة أقسام رئيسية:

1. فصائل النظام System Classes:

وهي المسئولة عن المهام العامة في التطبيقات مثل السرية Security وحفظ البيانات وتطبيقات الشبكات Networks والمهام المتعددة Multi Threading... إلخ.

2. فصائل النوافذ و الرسم Windows and Drawing Classes:

وهي المسئولة عن إنشاء الواجهة الرسومية للبرنامج Graphical User Interface (GUI) مع إمكانية الرسم علي النوافذ وإمكانية الطباعة:

3. فصائل البيانات Data Classes:

وهي المسئولة عن التعامل مع قواعد البيانات Databases لقراءة وكتابة البيانات في قاعدة البيانات Database.

4. فصائل الوب Web Classes:

وهي المسئولة عن إنشاء التطبيقات التي تعمل علي شبكة الإنترنت Internet.

## لغة وقت التنفيذ العامة Common Language Runtime :(CLR)

وهذه اللغة هي المسئولة أساساً عن تقديم بعض الخدمات عند تنفيذ البرنامج مثل:

1. إدارة الأوامر Code Management:

وهذه الخاصية تهتم بتحميل وتنفيذ Loading and Execution تطبيقات Visual Studio .NET حيث يتم تحويل الأوامر المكتوبة بلغة فيجوال بيسك Visual Basic إلي لغة وسيطة Intermediate Language (IL) ثم تتحول هذه اللغة إلي أوامر يمكن تنفيذها علي الجهاز الذي نريد تشغيل التطبيق عليه باستخدام المترجم اللحظي Just-In-Time (JIT) Compiler ، وبذلك لا نكون مقيدين بنظام تشغيل Operating System معين كما ذكرنا سابقاً.

2. دعم السرية Security Support:

حيث يمكن السماح بتشغيل جزء معين من الأوامر أو منع تنفيذ هذه الأوامر مثل محاولة كتابة أو قراءة بيانات سواء بالنسبة للجهاز الذي يتم تشغيل التطبيق عليه أو بالنسبة للشبكة التي يعمل عليها هذا الجهاز.

3. إخلاء الذاكرة Garbage Collection:

وهذه الخاصية مسئولة عن تتبع الذاكرة Memory وإخلائها من المتغيرات التي لسنا في حاجة إليها.

4. معالجة الأخطاء Error Handling:

وهذه الخاصية تمكننا من معالجة الأخطاء التي تظهر عند تنفيذ التطبيق مثل محاولة القسمة علي صفر ، وتتم معالجة هذه الأخطاء من خلال موضوع الاستثناءات Exceptions الذي سنقوم بشرحه في الفصول القادمة.

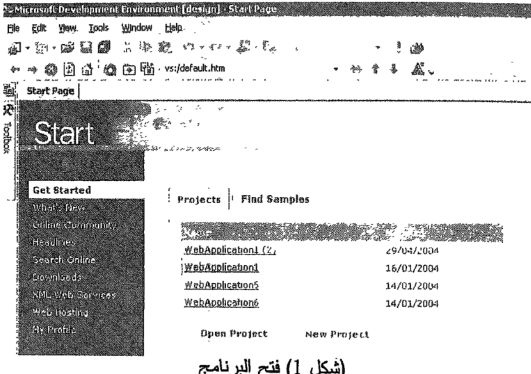
## تشغيل البرنامج:

سنقوم في النقاط القادمة بتشغيل البرنامج حتي نتعرف علي واجهة البرنامج كما سنقوم أيضاً بإنشاء أول تطبيق لنا في هذا الفصل لكي نتعرف أيضاً علي كيفية إنشاء التطبيقات في لغة فيجوال بيسك Visual Basic.

1. افتح البرنامج عن طريق اختيار

Start → Programs → Microsoft Visual Studio.Net → Microsoft Visual Studio.Net

لتظهر لك نافذة البرنامج كما هو واضح في شكل 1.

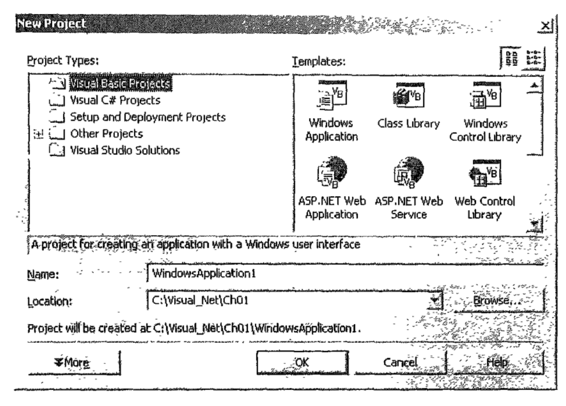



(شكل 1) فتح البرنامج

2. في هذه الشاشة تظهر لنا النافذة الافتتاحية للبرنامج حيث يمكننا من خلالها فتح مشروع Project سابق إنشائه أو إنشاء مشروع جديد.

3. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم

New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا الشاشة كما هو واضح في شكل 2.




شكل 2) إنشاء مشروع Project جديد

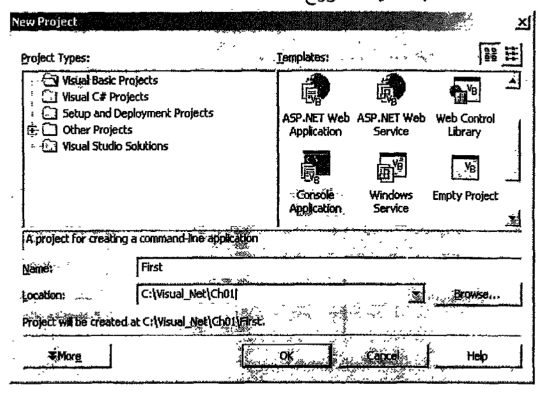
4. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".

5. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application" حيث أن هذا النوع من القوالب Templates يستخدم لإنشاء برنامج يقوم بإظهار نتائج عملياته في

شاشة الدوس Dos وسنقوم في الفصول القادمة بالتعرف علي بعض أنواع أخري من هذه القوالب Templates المتاحة.

6. في خانة الاسم "Name" نحدد اسما للمشروع وليكن First وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch01" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 3 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.

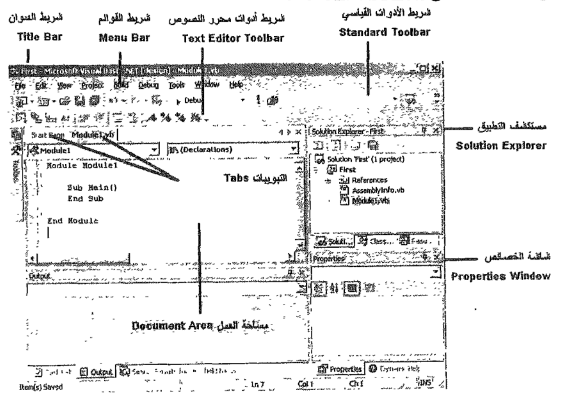



(شكل 3) إنشاء مشروع Project جديد

## Components of the IDE Window: مكونات واجهة البرنامج

تتكون واجهة البرنامج من المكونات التالية (انظر شكل 4)

1. شريط العنوان Title Bar.
2. شريط القوائم Menu Bar.
3. شريط الأدوات القياسي Standard Toolbar.
4. شريط أدوات محرر النصوص Text Editor Toolbar.
5. مستكشف التطبيق Solution Explorer.
6. شاشة الخصائص Properties Window.
7. التبويبات Tabs.
8. مساحة العمل Document Area.




شكل (4) مكونات واجهة البرنامج Components of the IDE Window

## أولاً: شريط العنوان Title Bar:

ويحتوي علي اسم المشروع Project الحالي وبجانب اسم المشـــروع Project يأتي اسم البرنامج [design] Microsoft Visual Basic .NET ويليـه اســم الملف الحالي والذي يكون اسمه الافتراضي هو Module1.vb.

## ثانياً: شريط القوائم Menu Bar:

ويحتوي علي جميع القوائم Menus المتاحة في البرنامج حيث تحتوي كل قائمة Menu علي مجموعة من الأوامر المرتبطة ، فمثلاً تحتوي القائمة Build علي مجموعة الأوامر الخاصة بترجمة الأوامر Build.

## ثالثاً: شريط الأدوات القياسي Standard Toolbar:

ويحتوي علي الأوامر الشائعة الاستخدام مثل أوامر فتح وحفظ وتنفيـذ الملـف. جميع الأوامر الموجودة في شريط الأدوات القياسي Standard Toolbar يمكن تنفيذها عن طريق القوائم Menus ولكن شريط الأدوات القياســي Standard Toolbar يعتبر أسرع وأسهل في الاستخدام.

ملحوظة:

إذا لم يكن شريط الأدوات القياســي Standard Toolbar ظــاهراً ، فيمكنـك إظهاره عن طريق فتح القائمة View ثم Toolbars ثم التأكد من وجود علامــة صح بجانب Standard ، فإذا لم تكن هناك علامة صح بجانــب Standard ، فيمكنك الضغط علي كلمة Standard ليــتم إظهــار شــريط الأدوات القياســي Standard Toolbar.

## رابعاً: شريط أدوات محرر النصوص Text Editor Toolbar:

ويحتوي علي الأوامر الخاصة بكتابة أوامر البرمجة. أيضاً جميع الأوامر الموجودة في شريط أدوات محرر النصوص Text Editor Toolbar يمكن تنفيذها عن طريق القوائم Menus ولكن شريط أدوات محرر النصوص Text Editor Toolbar يعتبر أسرع وأسهل في الاستخدام.

ملحوظة:

إذا لم يكن شريط أدوات محرر النصوص Text Editor Toolbar ظاهراً ، فيمكنك إظهاره عن طريق فتح القائمة View ثم Toolbars ثم التأكد من وجود علامة صح بجانب Text Editor ، فإذا لم تكن هناك علامة صح بجانب Text Editor ، فيمكنك الضغط علي كلمة Text Editor ليتم إظهار شريط أدوات محرر النصوص Text Editor Toolbar.

## خامساً: مستكشف التطبيق Solution Explorer:

ويظهر في الجزء الأيمن العلوي من نافذة البرنامج ويحتوي علي جميع الملفات التي يتم إنشاؤها في المشروع Project.

ملحوظة:

إذا لم يكن مستكشف التطبيق Solution Explorer ظاهراً ، فيمكنك إظهاره عن طريق فتح القائمة View ثم Solution Explorer.

## سادساً: شاشة الخصائص Properties Window:

وتظهر في الجزء الأيمن السفلي من نافذة البرنامج وتحتوي علي جميع خصائص الأدوات التي يتم إضافتها في المشروع Project ، وسنقوم باستخدامها لاحقاً في الفصول القادمة.

ملحوظة:

إذا لم تكن شاشة الخصائص Properties Window ظاهرة ، فيمكنك إظهارها عن طريق فتح القائمة View ثم Properties Window.

## سابعاً: التبويبات Tabs:

يقوم البرنامج بإنشاء تبويب Tab لكل ملف مفتوح حتي يمكنك فتح أكثر من ملف في نفس الوقت مع إمكانية التنقل بسهولة بين الملفات المفتوحة عن طريق الضغط علي تبويب Tab الملف المراد فتحه كما يمكنك أيضاً استخدام قائمة Window لتنفيذ نفس المهمة.

## ثامناً: مساحة العمل Document Area:

وهي تمثل مكان الأوامر المكتوبة.

تقوم لغة فيجوال بيسك Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة وسنقوم في الفصول القادمة بشرح هذا الكود ولكننا سنهتم هنا بكيفية تنفيذ الأوامر.

يمكنك تعديل كود البرمجة ليصبح كالتالي: (تم تظليل الأوامر الجديدة لسهولة التعرف علي مكانها)

```
1:    Module Module1
2:
3:        Sub Main()
4:            Console.WriteLine("This is my first project")
5:        End Sub
6:
7:    End Module
```

ملحوظات:

- لا تختلف الأحرف الكبيرة Capital عن الصغيرة Small في لغة فيجــوال بيسك Visual Basic.
- يقوم هذا البرنامج بطباعة رسالة علي شاشة الدوس Dos.
- لن نقوم بالشرح التفصيلي للأوامر المكتوبة في هذا الفصل حيث سنهتم هنــا بكيفية ترجمة Compile وتنفيذ الأوامر وسنقوم في الفصول القادمة بشرح هذه الأوامر فتابع التنفيذ.

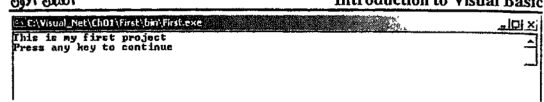
## تنفيذ المشروع Project:

يمكن تنفيذ المشروع Project بعدة طرق كالتالي:

1. فتح قائمة Debug ثم Start Without Debugging.
2. الضغط علي الزرين Ctrl + F5.
3. استخدام أداة التنفيذ المتاحة في شــريط الأدوات القياســي Standard Toolbar والتي تأخذ شكل علامة التعجب الحمراء.

يجب ملاحظة أن عملية تنفيذ البرنامج تقوم بحفظ الملف وترجمتــه Compile تلقائياً بدون أي تدخل من المبرمج ، وفي حالة حدوث أي أخطــاء فــي.كتابــة الأوامر ، فإن وصف الأخطاء يظهر في أسفل شاشة الأوامر.

- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علــي الشاشة كما هو واضح في شكل 5.




شكل (5 تنفيذ البرنامج)

- في هذه الشاشة يتم طباعة الجملة التي قمنا بكتابتها في الأوامر ويمكنك الضغط علي أي زر من لوحة المفاتيح Keyboard للرجوع إلي شاشة الأوامر.

## إغلاق المشروع Closing Solution:

بعد الانتهاء من إنشاء المشروع Project ، فإننا نريد أن نقوم بإغلاق المشروع Project تمهيداً لإنشاء مشروع جديد ، ويتم ذلك عن طريق فتح القائمة File ثم اختيار أمر Close Solution.

## إنهاء البرنامج Exit:

للخروج نهائياً من البرنامج ، فإنه يتم فتح القائمة File ثم اختيار أمر Exit (أو بالضغط علي الزرين Alt + Q).

## فتح المشروع Opening Solution:

إذا أردنا أن نقوم بالتعديل في المشروع Project الذي قمنا بإنشائه للتعديل فيه ، فتأكد من فتح البرنامج أولاً –انظر لبداية هذا الفصل– ثم افتح قائمة File ثم Open Solution ليتم فتح شاشة فتح المشروع Open Solution والتي سنقوم

فيها باختيار اسم ومسار المشروع Solution الذي نريد فتحه مع ملاحظــــة أن امتداد Extension ملفات المشروع Project يكون (.sln).

## ملخص الفصل:

تعلمنا في هذا الفصل أساسيات التعامل مـــع برنـــامج Visual Studio .NET وتعرفنا علي كيفية إنشاء مشروع Project بسيط يقوم بطباعة رسالة علي شاشة الدوس Dos كما تعرفنا علي واجهة البرنامج وكيفيـــة التعامـــل معهـــا وأهـــم مكوناتها.

# الفصل الثاني

## أنواع البيانات

## Data Types

في هذا الفصل نتعرف علـــي المتغيـــرات Variables والمؤثرات Operators والتعبيــرات Expressions وذلك من خلال النقاط التالية:

1. ما هي المتغيرات Variables.
2. أنواع المختلفة للبيانات Data Types.
3. المـــؤثرات الحســـابية Arithmetic Operators.
4. المؤثرات العلائقية والمنطقية Relational and Logical Operators.
5. مـــؤثر التخصـــيص Assignment Operator.
6. التعبير Expression.

أنواع البيانات
Data Types

## ما هى المتغيرات Variables:

المتغيرات هى مكان فى الذاكرة نستطيع تخزين معلومة فيه أثناء عمل البرنـــامج وهذا المتغير نستطيع تغيير قيمته فى أي وقت.
ونستطيع تخيل المتغير أنه عبارة عن وعاء لتخزين قيمة ما ونستطيع تغيير هذه القيمة وقتما نريد. والقيمة المخزونة فى هذا المتغير تتوقف علـــى نـــوع البيـــان Data Type الذى ينتمى إليه هذا المتغير فتعال معى نتعـــرف علـــي الأنـــواع المختلفة لهذه البيانات.

## الأنواع المختلفة للبيانات:

تنقسم أنواع البيانات فى لغة VB.net إلى فئتين رئيسيتين:

- بيانات مبنية ومعرفة فى أصل اللغة Built-in Data Types.
- بيانات يقوم المبرمج ببنائها (البيانات القائمـــة علـــى الفصـــائل Classes) وتسمي بأنواع المراجع Reference Types ، وسنتعرف على هذا النـــوع فى فصل البرمجة بالأهداف Object Oriented Programming.

والآن لنتعرف على النوع الأول:

- بيانات مبنية ومعرفة فى أصل اللغة Built-in Data Types.

وكما قلنا فهى البيانات التى تأتى مبنية فى أصل اللغة وهى تنقسم إلى اثنى عشر نوعاً أساسياً وذلك كما يوضحهم الجدول التالي حيث يتضح فيه نوع البيان Data Type والمساحة التى يحتلها فى الذاكرة وأيضا النطاق الرقمى الذى يستطيع كل نوع من الانواع استيعابه.

| النوع Data Type | المساحة فى الذاكرة | النطاق Range | |
|---|---|---|---|
| | | من | إلى |
| 1. البيانات الرقمية الصحيحة Integers Data | | | |
| Byte | 8 bits | 0 | 255 |

| 32767 | -32768 | 16 bits | Short |
|---|---|---|---|
| $2^{31}-1$ | $-2^{31}$ | 32 bits | Integer |
| $2^{63}-1$ | $-2^{63}$ | 64 bits | Long |
| 2. البيانات الرقمية ذات الفصلة العشرية Floating Data | | | |
| $3.4x10^{38}$ | $-3.4x10^{-38}$ | 32 bits | Single |
| $1.7x10^{308}$ | $-4.94065645841247 x10^{-324}$ | 64 bits | Double |
| $7.9x10^{28}$ | $10^{-28}$ | 128 bits | Decimal |
| 3. البيان الحرفي Character Data | | | |
| $2^{16}-1$ | 0 | 16 bits | Char |
| 4. البيانات المنطقية Boolean data | | | |
| false | true | 8 bits | Boolean |
| 5. البيانات الحرفية String | | | |
| أى حرفيات | | 16 bits | String |

## البيانات الرقمية الصحيحة Integers:

تحتوى البيانات الرقمية الصحيحة (أى الأرقام التى لا تحتوى على كسور) فـــى لغة VB.NET على أربعة أنواع وهم كما يوضحهم الجدول السابق كالآتى: Byte, Short, Integer, Long بالإضافة إلى Char ولكن النوع Char يمثـل أساساً الحروف.

## الإعلان عن المتغير Variable Declaration:

يتم الإعلان عن المتغير بكتابة الكلمة المحجوزة Dim ثم اسم المتغير ثم كلمــة AS ثم نوع المتغيرمع إمكانية إعطائه قيمة مبدئية Initial Value ، ولاختيــار اسم متغير ما ، فإن لك حرية تسمية المتغير على ألا يبدأ اسم المتغير بــرقم ولا يكون بين حروفه مسافة أو قوس أو الحروف الخاصــة مثــل (@ و # و...) ،

وفيما يلى أمثلة على الإعلان عن المتغيرات من النوع Integers:

```
Dim Salary As Integer=500
Dim C As Char="k"
Dim distance As Long
Dim Age As Byte age=30
```

وكما ترى فى الأمثلة السابقة أنه لكى نعلن عن متغير مثل Salary ، فنجد أنه قد سبقه الكلمة المحجوزة Dim ثم اسم المتغير وهو Salary ثم كلمة As ثم النوع الذى ينتمي إليه وهو Integer ، أما قيمة المتغير التى قمنا بتخزينها فى المتغير فهى 500 وذلك عن طريق المؤثر "=".

والمتغير Salary يحتل مساحة فى الذاكرة تقدر بـ 32 بت Bit لأنه من نوع Integer كما يتضح من الجدول السابق.

من الممكن الإعلان عن أكثر من متغير فى نفس الجملة كالتالى:

```
Dim x,y As Integer
```

ومن الأمور المتفق عليها (ولكنها ليست شرطاً) بين مطورى لغة VB.net فى تسمية المتغيرات أن يضاف مقطع من نوع البيان إلى اسم المتغير حتى يدل على نوعه أثناء قراءة الكود ودون الحاجة إلى الرجوع إلى السطر المعلن فيه وذلك كما فى الجدول التالى:

| نوع البيان | المقطع المضاف | مثال |
|---|---|---|
| String | str | strFirstName |
| Integer | int | intAge |
| Long | lng | lngLenghth |
| Double | dbl | dblThrustRatio |
| Boolean | b | bSex |

ملحوظات:

- إمكانية إعطاء قيم ابتدائية Initial Values للمتغيرات أثناء الإعلان عنها لم

تكن موجودة فى VB 6.0.

- إمكانية الإعلان عن أكثر من متغير فى نفس الجملة لم تكن موجودة فى VB 6.0.

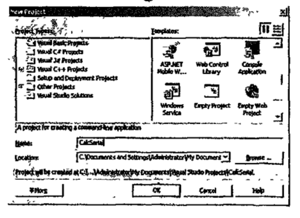
سنقوم الآن بكتابة برنامج يقوم بحساب مجموع سلسلة أرقام عدد معين (فمثلاً الرقم 4 يكون مجموع سلسلة أرقامه 1+2+3+4).

## مثال 1: المتغيرات Variables:

قم بتشغيل Visual Studio.net وذلك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

قم بإنشاء create مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع CalcSerial ثم اضغط الزر OK وذلك كما هو واضح فى الشكل التالى.



شكل 1 إنشاء مشروع Project جديد

عند الضغط على زر Ok تجد أن Visual Studio.net قد قام بكتابة الهيكل الرئيسى للبرنامج وذلك كما يماثل السطور التالية ماعدا المظلل منها الذى لابد أن تضيفه:

```
1.  Module Module1
2.  Sub Main()
3.  Dim i As Byte
4.  Dim SumSerials As Integer
5.  SumSerials = 0
6.  For i = 1 To 100
7.  SumSerials = SumSerials + i
8.  Next i
9.  Console.WriteLine("Summation of 100 is " &
    SumSerials)
10. Console.Read()
11. End Sub
12. End Module
```

## شرح الكود:

- فى السطر نخبر المترجم Compiler أننا نقوم بإنشاء وحدة Module جديدة اسمها Module1.
- فى السطر رقم 2 تجد الدالة Main() وهى الدالة الرئيسية داخل الوحدة التى أنشأناها وأى تطبيق من نوع Console لابد أن يحتوى على هذه الدالة Method.
- فى السطر رقم 3 أقوم بإنشاء متغير من نوع byte وقد أعطيته الاسم i.
- فى السطر رقم 4 أقوم بإنشاء متغير من نوع integer وقد أعطيناه الاسم SumSerials.
- فى السطر رقم 5 أقوم بشحن المتغير sumSerials بالقيمة 0.
- فى السطر رقم 6 أقوم بإنشاء دوارة Loop بحيث يكون عدد مرات تنفيذها هو 100 مع شحن المتغير i بالقيمة 1 عند إنشاء الدوارة Loop كما يتضح من السطر التالي:

For i=1 to 100

- فى السطر رقم 7 أقوم بتغيير قيمة المتغير SumSerials بقيمة المتغير i زائد القيمة الحالية للمتغير SumSerials فتعال معى نفهم كيفية عمل الدوارة Loop.

فى البداية تكون قيمة المتغير i=1 فإذا كانت قيمة المتغير i أقل من أو تساوى 100 يستمر فى العمل.

ثم يدخل فى السطر 7 فيقوم بجمع قيمة المتغير "i" والتى تساوى الآن القيمة 1 مع المتغير SumSerials الذى قيمته الحالية تساوي 0 ويتم تخزين القيمة فى المتغير SumSerials الذى أصبحت قيمته الآن 1.

ثم يتم تنفيذ جملة Next i التى تقوم بزيادة قيمة i بمقدار 1.

ثم بعد ذلك يرجع الى الدوارة loop مرة أخرى فتصبح قيمة المتغير i=2 وهى مازالت أقل من 100 فيجد أن الشرط مازال صحيحاً فيدخل الجملة SumSerials = SumSerials+I مرة أخرى فيقوم بإضافة قيمة المتغير i والتى تساوى 2 إلى قيمة المتغير SumSerials والتى تساوى 1 فتصبح قيمة SumSerials الآن تساوى 3.

ثم يدخل على Next i فتزيد قيمة i بمقدار 1 مرة أخرى.

فيتم الرجوع إلى الدوارة loop مرة أخرى وهكذا دواليك حتى تصبح قيمة i أكبر من 100 فيخرج من الدوارة loop ليتم تنفيذ الجملة التى بعدها.

فى السطر رقم 9 نجد الجملة المسئولة عن طباعة أى رسالة على الشاشة حيث يتم استخدام الفصيلة Console التى تحتوي بداخلها علي دالة الطباعة WriteLine() ، وعموماً لطباعة أى رسالة على الشاشة نستخدم الجملة التالية:

Console.WriteLine()

وما بين القوسين نكتب ما نريد إظهاره على الشاشة والجملة فى السطر 20

كانت كالآتي

```
Console.WriteLine("Summation of 100 is " & SumSerials)
```

ومعناها اطبع الجملة Summation of 100 is وقد أحطناها بعلامتي تنصيص Quotations كالتالي " " وذلك لأن أى حرفيات (سلسلة من الحروف) لابد وأن توضع بين علامتي تنصيص Quotations ، ثم بعد ذلك نريد طباعة القيمة الموجودة فى المتغير SumSerials فقمنا بوضع العلامة & ثم اسم المتغير ، والعلامة & تخبر المترجم بأن يقوم بعرض الجملة التالية للعلامة بالإضافة إلى الجملة أو الجمل السابقة.

ويمكن أن تحتوى الجملة على أكثر من جزء يراد طباعته كما فى الصورة التالية:

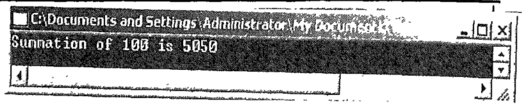
```
Console.WriteLine("My Name is "&Name&"My Salary
="&Salary)
```

ولاحظ أن اسم المتغير لم يوضع بين علامتي تنصيص Quotations لأنه متغير معروف عند المترجم Compiler.

ملحوظة:

عند عرض أى جملة على الشاشة باستخدام الجملة ()Console.WriteLine ، فلابد من وضعها بين علامتي تنصيص Quotations " ". أما المتغيرات فتكتب أسماؤها كما هى.

- فى السطر 10 نريد أن نجعل شاشة عرض النتيجة أن تنتظر فى حالة العرض حتي يمكنك رؤية نتيجة التنفيذ.
- قم بتنفيذ البرنامج وذلك عن طريق الضغط على الزرين Ctrl + F5 والشكل 2 يعرض نتيجة التنفيذ.




شكل 2 تنفيذ التطبيق

## البيانات الرقمية ذات الفصلة العشرية floating Point:

بينما تستطيع البيانات الرقمية integers تمثيل القيم الصحيحة فقط فإن البيانات ذات الفصلة العشرية floating Point تستطيع تمثيل القيمة التى تحتوى على قيم صحيحة وكسور مثل 39.99, -123.456.

ولغة VB.NET تشتمل على ثلاثة أنواع من بيانات الفصلة العائمة (العشرية) هم Single, Decimal & Double وفيما يلي أمثلة لهذه النوعية من البيانات:

```
Dim Salary as Decimal
Salary=3500.566
Dim Tax as Decimal
Tax=456789
Dim Rad as Double
Rad=0.56632
Dim SpaceLenghtUnit as Double
SpaceLenghtUnit=1452664554.5555545455
```

وبالطبع يتوقف اختيار كل نوع على القيمة التى سيتم تخزينها فى المتغير وعموماً اختر الأصغرمن أنواع البيانات Data Types كلما أمكن وذلك لعدم إهدار الذاكرة.

وعموماً فى لغة VB.NET نجد أن النوع double هو المستخدم بكثرة والسبب فى ذلك أن كثيراً من الدوال الحسابية والرياضية Math Functions تستخدم النوع double ، وسوف نستخدم الدالة Sqrt التى تقوم بحساب الجذر التربيعى فى المثال القادم.

## مثال 2: دالة الجذر التربيعي Sqrt():

قم بتشغيل Visual Studio.net وذلك عن طريق

Start ▶ Programs ▶ Microsoft Visual Studio.NET ▶ Microsoft Visual Studio.NET

قم بإنشاء مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع Sqrt ثم اضغط الزر OK.

ستجد أنه تم إنشاء كود مماثل للكود التالي ماعدا المظلل فقم بإضافته

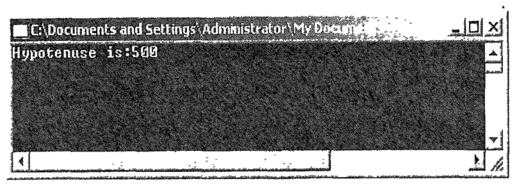
```
1.  Module Module1
2.  Sub Main()
3.  Dim x, y, z As Double
4.  x = 300
5.  y = 400
6.  z = Math.Sqrt(x * x + y * y)
7.  Console.WriteLine("Hypotenuse is:" & z)
8.  Console.Read()
9.  End Sub
10. End Module
```

### شرح الكود:

- سنقوم بشرح السطور المظللة حيث أن باقي الأسطر تماثل تماماً المثال السابق.
- في السطر 3 قمنا بالإعلان عن ثلاثة متغيرات من النوع double.
- في السطرين 4 و 5 قمنا بشحن المتغيرات x و y بالقيم 300 و 400 على الترتيب.
- في السطر 6 قمنا باستدعاء الدالة Sqrt وهي أحد دوال Method الفصيلة Math وهذه الدالة Method تقوم بإرجاع الجذر التربيعي لقيمة ما وهنا أعطيناها قيم $x^2$, $y^2$ (كضلعين في مثلث قائم الزاوية) فتقوم بحساب الجذر

التربيعى (لوتر المثلث) ثم يتم وضع الناتج فى المتغير z.

- فى السطر 7 نقوم بعرض قيمة المتغير z.
- فى السطر 8 نريد أن نجعل شاشة عرض النتيجة أن تنتظر فى حالة العرض لإعطاء الفرصة لمشاهدة شاشة نتيجة التنفيذ.
- قم بتنفيذ البرنامج وذلك عن طريق الضغط على الزرين Ctrl + F5 والشكل 3 يمثل البرنامج فى حالة التنفيذ.




شكل 3 تنفيذ التطبيق

## تشكيل الأرقام للعرض Formatting Numbers For Display:

عند التعامل مع البيانات الرقمية قد نضطر أحياناً إلى تشكيل الأرقام (القيمة التى يحتويها متغير ما) وتحويلها من صورة لأخرى وفى هذه الفقرة نتعرف على معرفة كيفية أداء ذلك.

### تحويل رقم إلى حروف:

فى بعض الأحيان نحتاج إلى تحويل قيم رقمية إلى حروف ويتم ذلك كالتالى:

```
Dim x As Integer
x=1500
x.ToString()
```

هنا تقوم الدالة ToString بتحويل قيمة المتغير x من قيمة رقمية إلى قيمة حرفية.

## استخدام الدالة ()Format:

تستخدم الدالة ()Format فى تشكيل التواريخ و الأرقام والحروف فمثلاً قد يرغب المحاسبون فى إظهار علامة العملة التى يستخدمونها بجانب الرقم أو تقسيم الرقم إلى مقاطع ووضع علامة ما مثل الفاصلة "," بين الأرقام.

والدالة ()Format تستقبل قيمة وحرف (أو حروف) تمثل رموز خاصة مبنية فى اللغة System Defined أو من الممكن أن تقوم بتشكيل رموز خاصة بك User-Defined والأمثلة التالية توضح استخدام الدالة ()Format.

## مثال 3: استخدام الدالة ()Format:

قم بتشغيل Visual Studio.net وذلك عن طريق

Start ►Programs ►Microsoft Visual Studio.NET► Microsoft Visual Studio.NET

ثم بإنشاء مشروع جديد من نوع Console Application ثم قم بكتابة السطرين التاليين داخل الدالة الرئيسية ()Main:

```
Dim Salary as Double=500.00
MsgBox(Format(Salary,"C"))
```

فى هذا المثال يتم عرض رسالة تحتوى على القيمة المخزنة فى المتغير Salary وبجانبها علامة العملة Currency حيث يرمز الحرف C إلى العملة (المعدة فى جهازك من خلال الإعدادات الإقليمية Regional Setting من لوحة التحكم Control Panel) وعند تنفيذ البرنامج تظهر الرسالة كما في الشكل التالي.



والسطور التالية توضح كيفية تشكيل رقم بما يلائم المستخدم.

قم بإضافة السطرين التاليين إلى نفس المشروع

```
Dim NetGross as Double=145055.33
MsgBox(Format(NetGross,"$#,###.00"))
```

حيث يمثل الرمز $ رمز العملة المراد استخدامها بينما الرمز # يحل محل أى رقم والشكل التالى يوضح نتيجة تنفيذ المثال حيث يتم ظهور قيمة المتغير Salary طبقاً لما حددناه فى الدالة Format().




والجدول التالى يوضح بعض الحروف المبنية فى اللغة التى تعمل على تشكيل الأرقام وتأثيرها.

| output | Input | الشرح | الرمز |
|---|---|---|---|
| 1,234.56$ | 1234.56 | يمثل عملة ويعمل هذا الرمز على طباعة الرقم مع وجود فاصلة ألفية (أي فصلة بعد كل 3 أرقام صحيحة) ورقمان بعد العلامة العشرية. | C |
| %1.23 | .0123 | يمثل علامة مئوية ويضرب الرقم فى 100 ثم يعرض الرقم متبوعا بعلامة %. | P |

| | | | |
|---|---|---|---|
| F | يقوم هذا الرمز بالعمل على عرض رقم واحد على الأقل يسار العلامة العشرية ورقمان فقط على يمين العلامة العشرية. | 1.5555 | 1.56 |
| X | يعمل هذا الرمز على تحويل الأرقام الصحيحة فقط إلى النظام السداسى عشر Hexadecimal. | 1234 | 2D4 |
| E | يعمل على إظهار الرقم فى شكل علامات أسية. | .0123 | 1.2300E-2 |
| N | يطبع الرقم بوجود علامات ألفية ويطبع رقم واحد على الأقل على يسار العلامة العشرية ورقمان على يمين العلامة العشرية. | 1234 | 1,234.00 |

حاول أن تستخدم الرموز الموجودة فى الجدول السابق مع الدالة Format().

## البيان الحرفي character:

غالباً فى برامجك سوف تحتاج إلى طريقة لتمثيل الحروف وليس فقط الأرقام والحروف هى أى رمز يستخدم فى الكتابة ومن أشهرها الحروف الأبجدية B,A Capital and Small كما يمكن معاملة الأرقام 0-9 أيضاً على أنها حروف ، وأيضاً علامات التعجب والمفاتيح الخاصة الموجودة فى لوحة المفاتيح Keyboard يمكن اعتبارها أيضاً حروف.

ولكى نعلن عن متغير من نوع حرف char يكون كالأتى.

Dim replay as Char="y"C

لاحظ أنك يجب أن تتبع قيمة المتغير بالحرف C (خاصة إذا كانت خاصية التحقق من نوع البيانات Type Check مضبوطة على Option Strict).

فى هذا المثال نحن نعلن عن متغير replay من نوع char ونخصص assign له القيمة y ويجب وضع القيمة بين علامتي تنصيص Quotations.

ملحوظة:

يجب وضع أى بيان من النوع char بين علامتي تنصيص Quotations " ".

ولإظهار قيمة هذا المتغير فى البرنامج فإننا نكتب الآتى:

Console.WriteLine("The Answer is:"& replay)

ويجب أن نلاحظ أن لغة VB.NET تستخدم نظام الـ Unicode فى تمثيل الحروف ولذلك نجد أن الحرف فى لغة VB.NET يتم تمثيله بعرض 16 بت Bit (Unsigned 16 bit) والذى يغطى النطاق من 0 الى 65535 كما يمكن أيضا استخدام النظام ASCII فى لغة VB.NET لأنه يعتبر مجموعة فرعية subset من الـ Unicode.

ملحوظة:

نظام الـ Unicode هو نظام تم ابتكاره ليستطيع تمثيل حروف أى لغة من لغات البشر.

## البيانات الحرفية String Data Type:

يستخدم هذا النوع فى تمثيل البيانات الحرفية التى تتكون من سلسلة من الحروف كالآتى:

Dim name As String ="Ahmed Osman"

ملحوظة:

- لتمثيل نصوص لابد من وضعها بين علامتي تنصيص Quotations " ".

## البيانات المنطقية bool Data Type:

إن القيمة التى يمكن ان يختزنها أى متغير من النوع Boolean تكون إما true أو false وهما كلمتان من الكلمات المحجوزة Reserved Words فى اللغة.

ملحوظة:

الكلمات المحجوزة Reserved Words هى كلمات تستخدمها لغة VB.NET ولانستطيع استخدامها فى تسمية المتغيرات أو أى عملية تخرجها عن نطاق عملها.

ونحتاج النوع bool لأننا فى أحياناً كثيرة أثناء البرنامج نحتاج لتحديد تحقق شرط ما.

وكمثال هل تحقق تنفيذ جزء من البرنامج أم لا ، ففى هذه الحالة نستطيع استخدام النوع boolean مثل:

```
Dim fileOpen As Boolean= true
```

وعموماً تستخدم البيانات من نوع bool مع الجمل الشرطية التى سنتحدث عنها لاحقاً.

وسنرى فى المثال التالى كيفية استخدام المتغيرات من النوع boolean.

## مثال 4: البيانات المنطقية boolean Data Type:

إذا لم تكن بيئة Visual Studio.net تعمل فقم بتشغيل Visual Studio.net وذلك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

قم بإنشاء مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع booleanType ثم اضغط الزر OK.

ستجد أنه قد تم إنشاء كود مماثل للكود التالى ماعدا المظلل فقم بإضافته

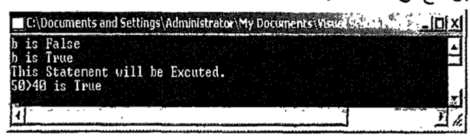
```
1.  Module Module1
2.  Sub Main()
3.  Dim b As Boolean
4.  b = False
5.  Console.WriteLine("b is " & b)
6.  b = True
7.  Console.WriteLine("b is " & b)
8.  If (b) Then
9.  Console.WriteLine("This Statement will be Excuted.")
10. End If
11. b = False
12. If (b) Then
13. Console.WriteLine("This Statement will not Excuted.")
14. End If
15. Console.WriteLine("50>40 is " & (50 > 40))
16. Console.Read()
17. End Sub
18. End Module
```

**شرح الكود:**

- فى السطر 3 قمنا بإنشاء متغير من النوع boolean اسمه b.
- فى السطر 4 قمنا بشحن المتغير بالقيمة (الكلمة المحجوزة) false.
- فى السطر 5 جملة عرض قيمة المتغيرb على الشاشة وسيطبع False.
- فى السطر 6 قمنا بشحن المتغير بالقيمة (الكلمة المحجوزة) true.
- فى السطر 7 جملة عرض قيمة المتغير b على الشاشة وسيطبع True.
- فى السطر8 أقوم بأختبار قيمة المتغير b ما إذا كانت true أم false ، فإذا كانت true فإنه يتم تنفيذ الجملة التى تليها فى السطر9 أما إذا كانت false فإنه لا يتم تنفيذ الجملة التى تليها. وبالطبع سيتم تنفيذ الجملة فى السطر 9 لأن قيمة b هى true.
- فى السطر 11 أقوم بتغيير قيمة المتغير b الى false.
- فى السطرين 12 و 13 هما نفس المفهوم الموجود فى السطرين 7 و8 مع

ملاحظة أن السطر 13 لن يتم تنفيذه لأن قيمة المتغير b هى false النـاتج عن السطر 11.

- فى السطر 15 أقوم بعرض نتيجة اختبار هل "40< 50" وطبعا لأن 50 أكبر من 40 يتم طباعة True.
- فى السطر 16 نريد أن نجعل شاشة عرض النتيجة أن تنتظر فى حالة العرض.
- قم بتنفيذ البرنامج وذلك عن طريق الضغط على المفتاح F5 و الشكل 4 يمثل البرنامج فى حالة التنفيذ.




شكل 4 تنفيذ التطبيق

## الثوابت الحرفية Literals:

**لفهم معنى الثوابت الحرفية أنظر معى عزيزى القارئ الى الجمل التالية**

1. Dim Salary as Integer=100
2. Console.WriteLine("My Salary is: “+salary)
3. Console.WriteLine("My Salary is:"+100)

كلا الجملتان 2 و 3 سيكون ناتج تنفيذهم بالصورة الآتية

My Salary is:100

والآن ما الفرق عزيزى القارئ بين الجملة 2 والجملة 3؟

الفرق عزيزى القارئ أنه فى الجملة رقم 2 يقوم المترجم بعـرض القيمـة 100 وهو يعلم يقيناً أنها من نوع integer وذلك نتيجة تنفيذ الجملة رقم 1 ويعلـم أن هذه القيمة يمكن أن تتغير.

أما فى الجملة رقم 3 فيقوم المترجم بالتعامل مع القيمة 100 بالطريقة التالية:

1. يقوم المترجم compiler بفحص القيمة ويحاول تخصيصها لنوع مـن أنواع البيانات بدءاً من أصغر الأنواع أي بدءاً من النوع byte ، فـإذا كانت القيمة أكبر من نطاق Range النوع فيقوم المترجم بالانتقال إلـى النوع الأكبر وهو Integer ثم long ثم double ثم Decimal.
2. يعامل المترجم القيمة 100 على أنها ثابت Constant أى أن قيمتها لن تتغير أبداً.

من هنا نستطيع وصف القيمة 100 بأنها ثابت حرفى Literal.

**إذا نستطيع تعريف الثابت الحرفى Literal بأنه قيمة ثابتة لا تتغير.**

## المؤثرات Operators:

المؤثر هو رمز خاص يستخدم فى العمليات الحسابية والمنطقية التى تجرى على المتغيرات.

## أنواع المؤثرات Operators Types:

توجد عدة أنواع من المؤثرات فى لغة VB.NET وهم:

1. المؤثرات الحسابية Arithmetic Operator.
2. المؤثرات المنطقية و العلائقية Logical & Relational Operators.
3. المؤثرات على مستوى البت Bit-Wise Operator.

## 1. المؤثرات الحسابية:

المؤثرات الحسابية وهى (^, Mod,/,-,+,*)

أمثلة على المؤثرات الحسابية:

| | |
|---|---|
| المؤثر (+) | Dim a as integer = 10<br>Dim b as integer =20<br>Dim Sum as integer<br>Sum = a + b; ' Sum =30 |
| المؤثر (−) | Dim salary as integer = 500<br>Dim deduction as integer = 85<br>int netSalary<br>netSalary = salary – deduction; |
| المؤثر (*) | Dim Salary as Integer = 500<br>Dim Bonus as Double = 2* Salary; |
| المؤثر (/) | Dim Salary as Integer = 500<br>Dim halfSal as Double=500/2 |
| المؤثر (Mod) وهو باقى القسمة Modulo | int x = 10Mod 3 ' x = 1 |
| مؤثر الزيادة والنقصان | وهو يعمل على زيادة المتغير بقيمة 1 فمثلاً كى نقوم بزيادة قيمة المتغير Count بالقيمة 1 كنا نقوم بعمل الآتى:<br>Dim count as Integer=0<br>count=count+1<br>ونستطيع فعل ذلك بطريقة مختصرة عن طريق المؤثر<br>count +=1<br>// doing the Same as count = count + 1<br>وتنطبق هذه الطريقة أيضا على المؤثر (−) الذى يعمل |

| على نقصان المتغير بقيمة 1. | |
|---|---|

## أمثلة مختلفة على اختصارات المؤثرات Operators:

يقوم هذا المثال بالإعلان عن متغير من نوع int ثم يقوم بإضافة 5.

```
Dim x as integer=10
x=x+5
```

يمكننا فعل ذلك بطريقة مختصرة كالآتي

```
x+=5
```

وينطبق ذلك على بقية المتغيرات

```
x=x*5
x*=5
x=x/3
x/=3
```

## 2. المؤثرات المنطقية و العلائقية Logical & Relational Operators:

المؤثرات المنطقية تمكننا من مقارنة حدين operands وكلا الحدين لابد وأن يكونا إما true أو false والمؤثرات المنطقية هى

(And,Or)

فتعال معى نفهم كل من هذه المؤثرات من خلال الجدول التالى حيث نرمز الى الحد الاول بحرف الـ P والحد الثانى بحرف الـ Q.

| not P | P ^ Q | P or Q | P And Q | Q | P |
|---|---|---|---|---|---|
| True | False | False | False | False | False |
| False | True | True | False | False | True |
| True | True | True | False | True | False |
| False | False | True | True | True | True |

تعال معى نفهم هذا الجدول

لو كان الحد P=false وكان الحد الثانى Q=false كما فى الصف المظلل يكون نتيجة استخدام المؤثرات كالتالى:

P AND Q =false
P Or Q= false

وهكذا ...مع بقية الجدول تبدأ بأخذ قيم P و Q من أول عمودين ثم تقوم بعد ذلك بمقارنة المؤثرات.

## a. المؤثر & (And)

هذا المؤثر يتطلب بأن يكون كلا الحدين true لكى يكون ناتج المقارنة true كما يتضح فى المثال التالى.

```
Dim x as Integer =10;
Dim y as integer =5
Dim s as integer=2
Dim t as integer t=3
if (x>y AND t>s) then
    'then do something
```

هذا الشرط سوف يتحقق لأن كل من التعبيرين صحيح true لأن الحد الأول "x>y" فيه قيمة x أكبر من قيمة Y وفى نفس الوقت الحد الثانى "t>s" فية قيمة T أكبر من قيمة s.

أما لو كان أحد الحدين غير صحيح false فإن الشرط بأكمله يصبح غير صحيح false كما فى المثال التالى

(4+3 = = 9) AND (3+3 = = 6)

فعلى الرغم من ان الحد الأيمن صحيح true نجد أن الحد الأيسر غير صحيح false فبالتالى يكون ناتج المقارنة false.

## b. المؤثر "or"

هذا المؤثر يتطلب فقط أن يكون أحد الحدين أو كلاهما true لكى يكــون نــاتج المقارنة true فمثلاً:

(3+6 = 5) OR (4+4 = 8)

ناتج المقارنة هنا يكون true لأن أحد الطرفين يرجع true لأن 8 = 4+4.

## c. المؤثر (XOR) EXCLUSIVE OR

هذا المؤثر يستخدم لتحديد ما إذا كان أحد الحدين فقط هو الصحيح true أما لــو كان كلا الطرفان false أو كلا الطرفان true فإن ناتج المقارنة يكون false.

وكمثال فإن ناتج المقارنة التالي يكون true

(5+7 =12)XOR (4+3=8)

لأن أحد الطرفين فقط هو الصحيح وهو(5+7 =12)

ولكن ناتج المقارنة التالي يكون false

(5+7 ==12) XOR(4+3==7)

لأن كلا الحدين true كما أن ناتج المقارنة التالية يكون false لأن كلا الحــدين FALSE

(5+7==13) XOR (4+3==8)

## d. المؤثر (not)

يقوم هذا المؤثر بعكس ناتج المقارنة

فمثلاً (4+3==5) يكون ناتج المقارنة false

ولكن مع استخدام المؤثر NOT.يصبح ناتج المقارنة true لأن المؤثر (NOT) قد قام بعكس ناتج المقارنة.

## 3. المؤثرات العلائقية Relational Operators:

وهذه المؤثرات تقوم بربط متغير بآخر واختبار العلاقة بينهما وهذه المؤثرات هى (=، > اقل من ،< اكبر من، => اقل من أو يساوى ، = < اكبر من أو يساوى، <> لا يساوى).

### المؤثر =

#### 1- المؤثر =

هذا المؤثر يقوم بتخصيص القيمة التى فى يمين العلاقة ووضعها فى المتغير الموجود فى يسار العلاقة فمثلاً

```
Dim x as Integer
     x=5
```

معنى هذا التعبير: قم بوضع القيمة 5 فى المتغير x.

#### 2- المؤثر ( = ) مع IF

هذا المؤثر يقوم على سؤال هل القيمة الموجودة فى يسار العلاقة تساوى الموجودة فى الناحية اليمنى أما لا فمثلاً

```
if ( x = 5)
```

والنتيجة تكون أما true أو false فإذا كانت true نقوم بكتابة ما نود فعله فى هذه الحالة أما لو كانت false فنقوم بكتابة ما نود عمله فى هذه الحالة الأخري.

وهكذا فباقى المؤثرات جميعها ترجع true أو false بناء على الشرط الذى تحدده ، وعموماً فإن هذه المؤثرات تستخدم مع الجمل الشرطية التى سنتعلمها فى الفصل الثالث.

والآن لنأخذ مثالاً على المؤثرات Operators.

## مثال 5: المؤثرات Operators:

إذا لم تكن بيئة Visual Studio.net تعمل قم بتشغيل Visual Studio.net

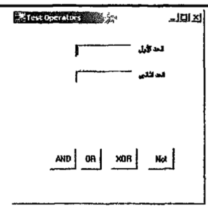
وذلك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

قم بإنشاء مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Windows Application وقم بتسمية المشروع LogicalOpTable ثم اضغط الزر OK.

قم بإضافة الأدوات الموجودة فى الجدول التالى عن طريق السحب والإفلات من المجموعة Windows Forms الموجودة فى صندوق الأدوات Tool Box الموجودة على يسار الشاشة ثم قم بعد ذلك بتغيير خصائص هذه الأدوات من نافذة الخواص window properties الموجودة على يمين الشاشة ليصبح شكل الفورم Form كما فى الشكل 5.

| قيمة الخاصية | الخاصية | الأداة |
|---|---|---|
| الحد الأول | Text | Label1 |
| الحد الثانى | Text | Lable2 |
| | Text | TextBox1 |
| | Text | TextBox2 |
| AND | Text | Button1 |
| ANDButton | Name | |
| OR | Text | Button2 |
| ORButton | Name | |
| XOR | Text | Button3 |
| XORButton | Name | |
| Not | Text | Button4 |
| NotButton | Name | |




شكل 5 إنشاء النموذج Form

ستجد أنه تم إنشاء كود مماثل للكود التالى ماعدا المظلل فقم بإضافته

```
1.  Public Class Form1
2.  Inherits System.Windows.Forms.Form
3.  Dim a, b As Boolean
4.  #Region " Windows Form Designer generated code "
5.  Public Sub New()
6.  MyBase.New()
7.  'This call is required by the Windows Form Designer.
8.  InitializeComponent()
9.  'Add any initialization after the InitializeComponent() call
10. End Sub
11. 'Form overrides dispose to clean up the component list.
12. Protected Overloads Overrides Sub Dispose(ByVal disposing As
    Boolean)
13. If disposing Then
14. If Not (components Is Nothing) Then
15. components.Dispose()
16. End If
17. End If
18. MyBase.Dispose(disposing)
19. End Sub
20. 'Required by the Windows Form Designer
21. Private components As System.ComponentModel.IContainer
22. 'NOTE: The following procedure_u105 ?s required by the Windows
    Form Designer
23. 'It can be modified using the Windows Form Designer.
24. 'Do not modify it using the code editor.
```

```
25. Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
26. Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
27. Friend WithEvents Label1 As System.Windows.Forms.Label
28. Friend WithEvents Label2 As System.Windows.Forms.Label
29. Friend WithEvents AndButton As System.Windows.Forms.Button
30. Friend WithEvents ORButton As System.Windows.Forms.Button
31. Friend WithEvents XORButton As System.Windows.Forms.Button
32. Friend WithEvents NotButton As System.Windows.Forms.Button
33. <System.Diagnostics.DebuggerStepThrough()> Private Sub
    InitializeComponent()
34. Me.TextBox1 = New System.Windows.Forms.TextBox
35. Me.TextBox2 = New System.Windows.Forms.TextBox
36. Me.AndButton = New System.Windows.Forms.Button
37. Me.Label1 = New System.Windows.Forms.Label
38. Me.Label2 = New System.Windows.Forms.Label
39. Me.ORButton = New System.Windows.Forms.Button
40. Me.XORButton = New System.Windows.Forms.Button
41. Me.NotButton = New System.Windows.Forms.Button
42. Me.SuspendLayout()
43. '
44. 'TextBox1
45. '
46. Me.TextBox1.Location = New System.Drawing.Point(96, 32)
47. Me.TextBox1.Name = "TextBox1"
48. Me.TextBox1.Size = New System.Drawing.Size(88, 20)
49. Me.TextBox1.TabIndex = 0
50. Me.TextBox1.Text = ""
51. '
52. 'TextBox2
53. '
54. Me.TextBox2.Location = New System.Drawing.Point(96, 72)
55. Me.TextBox2.Name = "TextBox2"
56. Me.TextBox2.Size = New System.Drawing.Size(88, 20)
57. Me.TextBox2.TabIndex = 1
58. Me.TextBox2.Text = ""
59. '
60. 'AndButton
61. '
62. Me.AndButton.Location = New System.Drawing.Point(56, 192)
```

```
63. Me.AndButton.Name = "AndButton"
64. Me.AndButton.Size = New System.Drawing.Size(40, 32)
65. Me.AndButton.TabIndex = 2
66. Me.AndButton.Text = "AND"
67. '
68. 'Label1
69. '
70. Me.Label1.Location = New System.Drawing.Point(192, 32)
71. Me.Label1.Name = "Label1"
72. Me.Label1.Size = New System.Drawing.Size(88, 24)
73. Me.Label1.TabIndex = 3
74. Me.Label1.Text = "الأول الحد"
75. '
76. 'Label2
77. '
78. Me.Label2.Location = New System.Drawing.Point(192, 72)
79. Me.Label2.Name = "Label2"
80. Me.Label2.Size = New System.Drawing.Size(88, 24)
81. Me.Label2.TabIndex = 4
82. Me.Label2.Text = "الثانى الحد"
83. '
84. 'ORButton
85. '
86. Me.ORButton.Location = New System.Drawing.Point(104, 192)
87. Me.ORButton.Name = "ORButton"
88. Me.ORButton.Size = New System.Drawing.Size(32, 32)
89. Me.ORButton.TabIndex = 5
90. Me.ORButton.Text = "OR"
91. '
92. 'XORButton
93. '
94. Me.XORButton.Location = New System.Drawing.Point(152, 192)
95. Me.XORButton.Name = "XORButton"
96. Me.XORButton.Size = New System.Drawing.Size(40, 32)
97. Me.XORButton.TabIndex = 6
98. Me.XORButton.Text = "XOR"
99. '
100.  'NotButton
101.  '
```

```
102.  Me.NotButton.Location = New System.Drawing.Point(208, 192)
103.  Me.NotButton.Name = "NotButton"
104.  Me.NotButton.Size = New System.Drawing.Size(40, 32)
105.  Me.NotButton.TabIndex = 7
106.  Me.NotButton.Text = "Not"
107.  '
108.  'Form1
109.  '
110.  Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
111.  Me.ClientSize = New System.Drawing.Size(292, 273)
112.  Me.Controls.Add(Me.NotButton)
113.  Me.Controls.Add(Me.XORButton)
114.  Me.Controls.Add(Me.ORButton)
115.  Me.Controls.Add(Me.Label2)
116.  Me.Controls.Add(Me.Label1)
117.  Me.Controls.Add(Me.AndButton)
118.  Me.Controls.Add(Me.TextBox2)
119.  Me.Controls.Add(Me.TextBox1)
120.  Me.Name = "Form1"
121.  Me.Text = "Test Operators"
122.  Me.ResumeLayout(False)

123.  End Sub

124.  #End Region

125.  Private Sub AndButton_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles AndButton.Click
126.  a = TextBox1.Text
127.  b = TextBox2.Text
128.  MsgBox(a And b)
129.  End Sub

130.  Private Sub orButton_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles ORButton.Click
131.  a = TextBox1.Text
132.  b = TextBox2.Text
133.  MsgBox(a Or b)
134.  End Sub
```

```
135.  Private Sub XorButton_Click(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles XORButton.Click
136.  a = TextBox1.Text
137.  b = TextBox2.Text
138.  MsgBox(a Xor b)
139.  End Sub

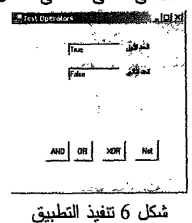
140.  Private Sub NotButton_Click(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles NotButton.Click
141.  a = TextBox1.Text
142.  MsgBox(Not a)
143.  End Sub
144.  End Class
```

## شرح الكود:

- فى السطر 3 قم بالإعلان عن متغيرين a,b من نوع Boolean وهما المتغيران اللذان سنقارن قيمتهما باستخدام المؤثرات Operators.
- قم بالذهاب إلى النموذج Form (النافذة التى ظهرت بعد إنشاء المشروع) ثم قم بالضغط مرتين Double-click على الزر AND وستجد أنك انتقلت إلى نافذة الكود وهذا الكود يعالج حدث الضغط Click Event أى ما يتم تنفيذه من الكود عند الضغط على الزر ثم قم بكتابة السطور 126و127و128 وهذه الجمل تؤدى الوظائف التالية.
- السطر 126 نقوم فيه باستقبال القيمة التى سيدخلها المستخدم عن طريق TextBox1.Text أى النص المدخل فى أداة النص TextBox1 ثم يتم بتخزين هذة القيمة فى المتغير a الذى قمنا بالإعلان عنه فى السطر3.
- السطر 127 يماثل السطر 126 فيما عدا أنه يعمل مع أداة النص الثانية ويقوم بتخزين القيمة المدخلة فى المتغير b.

- في السطر 128 نستدعي الدالة MsgBox() وهي دالــة Method تعمــل على ظهور صندوق رسالة Message Box تحمل نصاً وهذا النص هو ما يكتب ما بين القوسين ، وستجد بين القوسين أننا قد استخدمنا المؤثر AND بين المتغير a,b وسيتم التنفيذ بناء على قيمة كل منهم ويتم إرجاع قيمة إمـــا True أو False تظهر فى صندوق الرسالة.
- قم بالذهاب إلي النموذج Form مرة أخرى ثم قم بالضغط مرتين -Double click على الزر OR ثم قم بكتابة السطور 131و132و133 و هى تعمــل بنفس الطريقة التى يعمل فيها الكود فى الزر AND فيما عـــدا أننـــا هنـــا نستخدم المؤثر OR.
- قم بالذهاب إلي النموذج Form مرة أخرى ثم قم بالضغط مرتين -Double click على الزر XOR ثم قم بكتابة السطور 136و137و138.
- قم بالذهاب إلي النموذج Form مرة أخرى ثم قم بالضغط مرتين -Double click على الزر NOT ثم قم بكتابة السطور 141و142.

قم الآن بتنفيذ البرنامج عن طريق الضغط علي الزر F5 ثم قم بكتابة True فى صندوق الكتابة الأول و False فى الثانى كما فى الشكل التالى.




شكل 6 تنفيذ التطبيق

ثم قم بضغط الأزرار وانظر ماذا ترى! حاول تبديل وتغيير القيم فى كـــل مـــرة وستجد أن النتائج تماثل ما قمنا بشرحه سابقاً.

## أسبقية تنفيذ المؤثرات Operator Precedence:

ماهو المؤثر الذى سيتم تنفيذه أولا من خلال المثال التالى؟

(3<5) AND (2=1)OR(7=7)

والآن هل هذه العلاقة true أم false؟ أى هل يــتم تنفيــذ المــؤثر OR أولاً أم المؤثر AND ؟؟

والحل هو أن هذه العلاقة true ولكن لماذا؟!

لأن المؤثر AND له أسبقية فى التنفيذ عن المؤثر OR ولكى تحل هذا المثــال بشكل صحيح تعال نضع أنفسنا مكان المترجم Compiler ولنرى كيف ســيقوم بالتنفيذ.

1- فى البداية سيقوم بتنفيذ العلاقة التى بها المؤثرAND ويكون الشكل كالتالى.

((3<5) AND (2=1)) OR (7=7)

ولأن 3 أقل من 5 فالعلاقة true و2 لا تساوى 1 فالعلاقة false

((true) AND (false)) OR (7=7)

وبعد ذلك يجئ دور المؤثر AND الذى يقوم بالمقارنة بين true ، false فتكون النتيجة false ويصبح شكل العلاقة كالتالى.

false OR (7=7)

ولان العلاقة7=7 تنتج true فيكون شكل العلاقة

false OR true

وبالطبع لأن أحد طرفى العلاقة هو true والمؤثر هو ، فيصبح ناتج المقارنــة كلها true.

### وأسبقية تنفيذ المؤثرات نرى ترتيبها من خلال الجدول التالى

| المؤثر | أهمية التنفيذ |
|---|---|
| () [] . checked new sizeof typeof unchecked | 1 |
| ! ~ (cast) +(unary) -(unary) | 2 |

| | |
|---|---|
| 3 | * / % |
| 4 | + - |
| 5 | < > <= >= is |
| 6 | = <> |
| 7 | AND |
| 8 | ^ |
| 9 | OR |

كما ترى فإن بعض هذه المؤثرات لم نتعرض له والبعض الأخر تعاملنا معه ، وطبقاً لهذا الجدول فإن أى أقواس "()" لها أسبقية قصوى فى التنفيذ.

## التعبيرات Expressions:

إن التعبيرات فى لغة VB.NET تتكون من خلط المتغيرات variables مع بعضها البعض باستخدام المؤثرات operators والثوابت الحرفية Literals وقد تعاملنا بالفعل مع التعبيرات وكمثال:

```
1. Dim bonus as Integer=100    'statement
2. Dim salary as Double=3500.5   ' statement
3. Dim NetSalary As Double=Salary+bonus-50
```

نجد أن السطر 3 هو تعبير Expression وذلك لأننا استخدمنا المتغيرات مع المؤثرات مع الثوابت الحرفية.

وكما نرى فى السطر 3 قمنا بخلط أنواع مختلفة من البيانات مع بعضها ويتعامل معها المترجم بأن يقوم بتحويل convert البيانات المختلفة الى بيان من نوع واحد فى النهاية .

## التعامل مع الوقت و التاريخ:

توفر لغة VB.net أنواع عديدة من أنواع البيانات لتخزين الوقت والتاريخ كما فى الأمثلة التالية:

- Dim dtDate1 As Date= # 12/77/1974 #

- Dim dtKing As Date=Convert.TODateTime(" Aug 16,1977 5:00 PM")
- Dim dtJFK as Date=Convert.ToDateTime(" December 23, 1963")
- Dim dtMoon as New Date(1969,7,20)

في السطر الأول من الأمثلة السابقة استخدمنا فيه العلامة # لتخصيص قيمة للمتغير dtDate1 وفي الجملتين التاليتين استخدمنا فيهما الدالة Convert.TODateTime() التي تقوم بتحويل النص إلى تاريخ ووقت. أما الجملة الأخيرة فتستخدم الفصيلة Date Class (سنتعرف على معنى الفصائل Classes وكيفية بنائها في الفصول القادمة).

يجب أن تأخذ فى اعتبارك أن نوع البيانات من نوع Date دائماً يحتوى تاريخ ووقت معاً ممثلاً بالمللى ثانية Milli Second. وتوجد العديد من الدوال Methods فى نوع البيانات Date لاستخلاص أجزاء الوقت و التاريخ مثل اليوم و الشهر و السنة.

## تحديد الوقت و التاريخ الحالي Determining The Current :Date And Time

أحيانا نواجة مواقف لكيفية تحديد الوقت أو التاريخ الحاليين. وتوجد دالتان لمعرفة الوقت و التاريخ الحاليين هما Now() و Today() ، فالدالة Now() تقوم بإرجاع الوقت و التاريخ الحاليين.

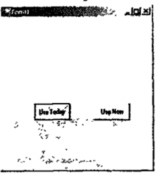
## مثال 6: الوقت و التاريخ :Date And Time

إذا لم تكن بيئة Visual Studio.net تعمل قم بتشغيل Visual Studio.net وذلك عن طريق

Start ►Programs ►Microsoft Visual Studio.NET► Microsoft Visual Studio.NET

قم بإنشاء مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Windows Application وقم بتسمية المشروع UseDateTime ثم اضغط الزر OK.

قم بإضافة زرين للنموذج Form الحالي واجعل شكله كما فى الشكل التالى:



شكل 7 نموذج Form الوقت والتاريخ

ثم قم بالضغط مرتين Double-click على الزر Use Now واكتب الكود التالى:

```
MsgBox(Date.Now)
```

ثم قم بالضغط مرتين Double-click على الزر Use Today واكتب الكود التالى:

```
MsgBox(Date.Today)
```

قم الآن بتنفيذ البرنامج عن الطريق الضغط علي الزر F5 واضغط الزرين ولاحظ الفرق بينهما!

## تشكيل الوقت والتاريخ Date And Time Format:

نحتاج عند التعامل مع الوقت والتواريخ إلى تشكيله بما يتناسب مع رغباتنا. فمثلاً عند طباعة التقارير نحتاج مثلا أن يكون التاريخ قصيراً بمعنى وجود رقمين لليوم ورقمين للشهر ورقمين للسنة ، وتمدنا لغة VB.net بحروف تمكننا عند وضعها من تشكيل التاريخ بناء على شكل هذه الحروف. والجدول التالى يمثل هذه الحروف وأمثلة تمثل شكل التاريخ عند استعمال هذه الحروف.

| حروف التشكيل Formating String | مثال لشكل التاريخ |
|---|---|
| M/d/yy | 8/16/45 |
| MM/dd/yyyy | 08/16/1945 |
| MM/dd hh:mm:mm:ss tt | 08/16 01:15:00 pm |
| MMMM d, yyyy H:mm | August 16, 1945 13:15 |
| dddd, mmm d, yyyy | Thursday, Aug 16, 1945 |

ولمزيد من الفهم تابع المثال التالى.

قم بإضافة زر Button إلى النموذج Form الموجود فى المثال السابق وقم بتغيير الخاصية Text له من شاشة الخصائص Properties Window واجعلها ShowFormatDate ثم قم بالضغط عليه مرتين Double-click وستجد نافذة الكود قد ظهرت فقم بكتابة المظلل الكود التالى:

```
Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
1.   Dim TheDate As Date = Now
2.   Dim OutPutString1 As String
3.   OutPutString1 = Format(TheDate, "M/d/yy")
4.   MsgBox(OutPutString1)
End Sub
```

- فى السطر الأول قمنا بالإعلان عن المتغير TheDate من نوع Date وقد خصصنا assign الدالة Now (التى تقوم بإرجاع التاريخ الحالى) فيصبح المتغير TheDate يحمل التاريخ الحالى.
- فى السطر الثانى قمنا بالإعلان عن متغير OutPutString من نوع String والغرض منه احتواء المتغير TheDate بعد إجراء التشكيل Format عليه باستخدام الدالة Format().

- فى السطر 3 نستدعى الدالة Format() ونرسل لها معاملين Parameters الأول هو التاريخ المراد تشكيله TheDate و الثانى الحروف المراد تشكيل التاريخ على أساسها وهى "M/d/yy".
- فى السطر 4 نظهر التاريخ باستخدام الدالة MsgBox() ونقـوم بطباعـة المتغير OutPutString1.
- قم الآن بتنفيذ البرنامج عن طريق الضغط علي الزر F5 ثم اضغـط الـزر ShowFormatDate و انظر ماذا ترى.
- قم بتغيير التنسيقات المختلفة للتاريخ طبقاً للجدول السابق ولاحظ التغيير فى كل مرة.

## العمل مع أجزاء من التاريخ Extracting Parts OF Date:

فى بعض المواقف نحتاج إلى معرفة جزء صغير من التاريخ مثل اليوم فقـط أو الشهر أو السنة أو الوقت فقط أو الساعات أو...

وتمدنا لغة VB.net بعدد من الخواص Properties للمتغيرات التى مـن نـوع Date سنوضحها فى الجدول التالى:

| أسم الخاصية | المعلومة الناتجة عن الخاصية | نوع البيانات المرتجع |
|---|---|---|
| Date | تاريخ فقط | Date |
| TimeOFDay | وقت فقط | TimeSpan |
| Month | شهر من(1-12) | Intgere |
| Day | يوم من الشهر من (1-31) | Integer |
| Year | سنة | Integer |
| Hour | ساعة من(0-23) | Integer |
| Second | ثانية من(0-59) | Integer |
| Millisecond | مللى ثانية (0-999) | Integer |

| Integer | رقم اليوم فى الأسبوع(0-6) | DayOfWeek |
|---|---|---|
| Integer | رقم اليوم فى السنة (0-365) | DayOfYear |
| Long | التكات منذ 0000/1/1 | Ticks |

وكمثال لاستخدام هذه الخواص قم بإضافة زر Button إلى المشروع السابق وقم بتغيير خاصية النص Text إلى ShowTheYear ثم قم بالضغط على الزر مرتين Double-click لتنتقل إلي كود الأوامر فاكتب الكود المظلل

```
Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
1.      Dim TheDate As Date = Now
2.      MsgBox(TheDate.Year)
End Sub
```

تجد هنا أننا استخدمنا الخاصية Year ويمكنك تغييرها بباقى الخواص الموجودة فى الجدول السابق.

## العمل مع الحرفيات Working with Strings:

من أهم المهارات بالنسبة للمبرمج هو قدرته على العمل مع أنواع البيانات وتطويعها ومعرفة خصائصها ومن أهم هذة البيانات هو البيانات الحرفية Strings ؛ ويوجد نوعان من البيانات الحرفية فى لغة VB.net هم Char,String وكما تعرفنا سابقاً أن النوع Char يستطيع حمل حرف واحد فقط ويكون شكل إنشاء متغير منه كالتالى:

```
Dim PSex as Char
PSex="M"
```

أما النوع String فيستطيع العمل مع أى عدد من الحروف

```
Dim MyName As String="Kadry Hessien"
```

## وصل الحرفيات String Concatenation:

من المواقف التى يواجها المبرمجون أحياناً هو وصل الحرفيات كما يتضح فـى المثال التالى:

```
Dim FirstName As String="Kadry"
Dim LastName As String="Hessien"
```

من الواضح أنه يوجد متغيران كل منهم يحمل قيمة والسؤال المطروح هو كيـف نضم قيمة المتغيران فى متغير واحد؟ لحل هذا نستخدم المؤثر & كالآتى:

```
Dim FullName as String
FullName=FirstName & LastName
```

فتصبح قيمة المتغير FullName الآن هى "KadryHessien".

كما يمكننا أيضا فعل ذلك باستخدام الدالة String.Concat() كالآتى:

```
FullName=String.Concat(FirstName,LastName)
```

كما يمكننا وضع مسافة بينهم لفصل الاسمين عن بعضهما البعض كالتالى:

```
FullName=String.Concat(FirstName, " ", LastName)
```

## تحديد طـول الحرفيـات Determining the Length of The String:

قد تحتاج فى بعض المواقف إلى معرفة عدد الحروف المكونة لمتغير ما من نوع String. ولمعرفة عدد الحروف نستخدم الخاصية Length كمـا فـى المثـال التالى:

```
Dim FullName as String="Kadry Hessien"
Dim MyNameLenght as Integer
MyNameLenght= FullName.Length
```

وتكون قيمة المتغير MyNmaeLength هي 13.

## تغيير حالة الحروف Changing the case of a String:

من الأمور المفيدة أحياناً تغيير حالة الحروف من كبيرة Capital إلى صـــغيرة Small والعكس ويتم ذلك باستخدام الدالة ToUpper() كالتالى:

```
Dim FullName as String="Kadry Hessien"
FullName.ToUpper()
```

فيصبح قيمة المتغير FullName هى "KADRY HESSIEN".

وأظنك قد خمنت العكس وهو كالأتى

```
FullName.ToLower()
```

فتصبح قيمة المتغير FullName هى "kadry hessien".

## البحث عن حرفيات Searching for Strings:

أحياناً قد تحتاج إلى البحث عن حرف أو حروف فى سلسة حروف. وتوجد دالة هى IndexOf() وهى تقوم بإرجاع رقم (ترتيب) أول حرف يتطابق مع الحرف أو الحروف التى تبحث عنها إن وجدت كما فى المثال التالى:

```
Dim StrLine as String="I go to School every Day"
Dim x as Integer
X=StrLine.IndexOF("to")
```

فى هذا المثال أعلنا عن المتغير StrLine من String وأعطيناه القيمة

"I go to School every Day"

وأردنا أن نبحث عن كلمة "to" هل هى موجودة فى المتغير StrLine أم لا؟

لذلك استخدمنا الدالة IndexOf() التى تقوم بالبحث فى المتغير StrLine وفـــى هذه الحالة تقوم بارجاع القيمة 5 (ترتيب الحرف t والعد يبدأ من صفر).

# الفصل الثالث

## جمل التحكم

## Control Statements

جمل التحكم Control Statements

في هذا الفصل نتعرف علي كيفية استخدام جمل التحكم Control Statements والدوارات Loops وذلك من خلال النقاط التالية:

1. استخدام if و for.
2. استخدام select..case.
3. استخدام while.
4. استخدام do...while.

**مقدمة:**

فى هذا الفصل سوف نشرح جمل التحكم والتى مهمتها التحكم فى انسياب تنفيذ البرنامج program flow execution ، ونستطيع تقسيم جمل التحكم فى البرنامج الى ثلاثة فئات وهم:

- جمل الاختيار (الشرط) selection statement والتى تشمل if, select .
- جمل التكرار iteration statements والتى تشمل for ,while ,do-while.
- جمل القفز(كسر التكرار أو السيطرة) jump statement والتى تشمل break ,continue , goto ,return.

## الجمل الشرطية Conditional Statements

### الجملة الشرطية if

أول شكل من أشكال الجملة الشرطية if يأخذ الصورة العامة التالية:

```
IF(condition) Then
statement
End IF
```

**شرح الصورة:**

تقوم الجملة الشرطية if باختبار الشرط الذى بين القوسين () فإذا تحقق يتم تنفيذ الجملة التالية لجملة if وإذا لم يتحقق ينتقل مسار البرنامج الى الجملة التى تلى END IF ثم يتم تنفيذ الجملة التى تلى else كما فى المثال التالى

```
Dim Salary As Integer=3000
IF (Salary>2000) Then
Console.WriteLine("wow great Salary")
End IF
```

فى هذا المثال تقوم الجملة if باختبار المتغير Salary ما إذا كان أكبر من 5000 أم لا؟

فإذا كان أكبر (وهو كذلك) من 5000 يقوم البرنامج بطبع الرسالة

wow great Salary

ويجب أن تلاحظ عزيزى القارئ أن جملة IF يجب أن تنتهى بـEnd IF .

فى المثال السابق قد يبدو هناك سؤال وهو ماذا لو أردنا أن نفعل شئ عند عدم تحقق الشرط ؟! هنا يجئ دور Else ومعناها عند عدم تحقق شرط IF نفذ الجملة(أو الجمل) التى تلى Else وتكون كما فى المثال التالى:-

```
1. Dim Salary As Integer=1000
2. IF (Salary>2000) Then
3. Console.WriteLine("wow great Salary")
4. Else
5. Console.WriteLine("You Need a new Job")
6. End IF
```

فى هذا المثال نجد أن شرط جملة (Salary>2000) IF لن يتحقق لأن المتغير Salary أقل من 2000 وبالتالى لن يتم تنفيذ السطر 3 وسيتم تنفيذ Else أى أنة سيتم تنفيذ السطر 5 .

## مثال 1: الجملة الشرطية if:

قم بتشغيل Visual Studio.net وذلك عن طريق

Start Programs ▶ Microsoft ▶ Microsoft Visual Studio.NET ▶ Visual Studio.NET

أما إذا كان Visual Studio.net يعمل فقم بالذهاب الى قائمة

File▶ Close Solution

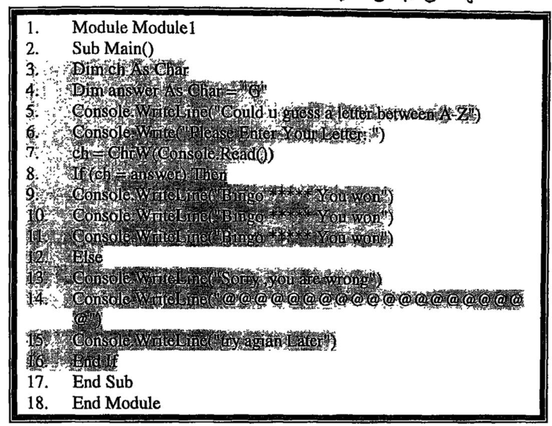
ثم

File▶New▶Project

قم بإنشاء create مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع ControlIf

ثم اضغط الزر OK.

1. عند الضغط على زر Ok تجد أن Visual Studio.net قد قام بكتابة الهيكل الرئيسى للبرنامج وذلك كما يماثل السطور التالية ما عدا المظلل منها الذى لابد أن تضيفه.



```
1.  Module Module1
2.  Sub Main()
3.  Dim ch As Char
4.  Dim answer As Char = "G"
5.  Console.WriteLine("Could u guess a letter between A-Z")
6.  Console.Write("Please Enter Your Letter: ")
7.  ch = ChrW(Console.Read())
8.  If (ch = answer) Then
9.  Console.WriteLine("Bingo ***** You won")
10. Console.WriteLine("Bingo ***** You won")
11. Console.WriteLine("Bingo ***** You won")
12. Else
13. Console.WriteLine("Sorry ,you are wrong")
14. Console.WriteLine("@@@@@@@@@@@@@@@@@@@
    @")
15. Console.WriteLine("try agian Later")
16. End If
17. End Sub
18. End Module
```

## شرح الكود:

الهدف من البرنامج هو لعبة التخمين حيث يخمن المستخدم حرفاً وإذا كان صحيحاً يتم إظهار رسالة وإذا كان خطأ يتم إظهار رسالة تفيد ذلك.

- فى السطر 3 قمنا بالإعلان عن المتغير ch والذى نخزن فيه الحرف الذى سنستقبله من المستخدم.
- فى السطر 4 قمنا بالإعلان عن المتغير answer والذى سيحتوى القيمة G التى نقارن معها القيمة التى سنتلقاها من المستخدم.

- فى السطرين 5 و 6 نقوم بتوجيه رسالة إلى المستخدم لمحاولة تخمين الحرف.
- فى السطر 7 نقوم باستقبال الحرف من المستخدم ويتم تخزينه فى المتغير ch باستخدام الدالة Read() وهى الدالة التى تقرأ القيم الأتية من لوحة المفاتيح. وتستقبل هذه الدالة القيم الأتية كأرقام ولذلك كان يجب تحويل هذه القيم إلى قيم من نوع char لذلك إستخدمنا الدالة CHrw.
- فى السطر 8 نقوم باختبار قيمة الحرف الذى تم إدخاله (هل يساوى القيمة المخزونة فى المتغير answer والتى تساوى G).
- فإذا كان الحرف الذى تم إدخاله هو الحرف G بحرف كبير Capital فإننا نريد إظهار ثلاث جمل كما يتضح من خلال السطور من 9-11.
- أما إذا كان الحرف الذى تم إدخاله ليس G فإننا نريد أن نظهر الجمل التى فى السطور من 13-15 ولذلك قمنا بكتابتهم داخل Else.

وبعد أن رأينا عزيزى القارئ الجملة الشرطية if بصورها المختلفة تعال معى نتعرف على جملة Select.

## :Select Case جملة الشرط

هذه الجملة تتشابه فى عملها مع الجملة الشرطية if-else-if وتستعمل فى اختبار قيمة ضد قيم أخرى ثابتة ، فمثلاً لو كان هناك تقدير مثل ( مقبول و جيد و...) وهذا التقدير يتوقف على قيمة درجة الطالب ، فنستطيع استخدام الجملة الشرطية Select. والجملة الشرطية Select تأخذ الشكل التالى:

```
Select case(var)
case value1
do result one
```

```
case value2
do result two

.
Case Else
do default result
End Select
```

في هذه الصورة:

- نبدأ بالكلمة switch وبداخلها المتغير var الذى يتم اختباره ثم تبدأ الحالات المتوقعة لقيمة هذا المتغير var.
- الحالة الأولى case value1 تقول أنه فى حالة تساوي قيمة المتغير var مع القيمة value1 ، فقم بتنفيذ النتيجة الأول do result one وهكذا.
- وفى النهاية نجد جملة Case Else التى تنفذ فى حالة عدم تحقق أى حالة مع ملاحظة أن الكلمات المحجوزة reserved Words فى هذه الصورة هى:

Select,Case,Case else,End Select

- o الكلمة ()Select case لاختبار قيمة المتغير.
- o الكلمة case لتحديد الحالات المختلفة للمتغير.
- o الكلمة Case Else لتنفيذ مجموعة من الجمل فى حالة عدم تحقق أى حالة.

كما لا يسمح أن تكون هناك جملتين متماثلتين من جمل case مثل:

```
Select Case x
case 0
statement;
case 0
statement;
.
```

End Select

هذا الشكل غير مسموح به (كما أنه غير منطقى).

ولنأخذ مثالاً على الجملة الشرطية switch.

## مثال 2: جملة switch:

إذا لم يكن Visual Studio.net يعمل قم بتشغيل Visual Studio.net وذلك عن طريق

Start ►Programs ►Microsoft Visual Studio.NET► Microsoft Visual Studio.NET

أما إذا كان Visual Studio.net يعمل فقم بالذهاب الى قائمة

File► Close Solution

ثم

File► New► Project

قم بإنشاء create مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع SelectCase ثم اضغط الزر OK.

عند الضغط على زر Ok تجد أن Visual Studio.net قد قام بكتابة الهيكل الرئيسى للبرنامج وذلك كما يماثل السطور التالية ما عدا المظلل منها الذى لابد أن تضيفه.

```
1.   Module Module1
2.   Sub Main()
3.   Dim i As Integer
4.   For i = 0 To 10
5.   Select Case i
6.   Case 0
7.   Console.WriteLine("i is Zero")
8.   Case 1
```

```
9.  Console.WriteLine("i is one")
10. Case 2
11. Console.WriteLine("i is two")
12. Case 3
13. Console.WriteLine("i is three")
14. Case 4
15. Console.WriteLine("i is four")
16. Case Else
17. Console.WriteLine("i is five or more")
18. End Select
19. Next i
20. Console.Read()
21. End Sub
22. End Module
```

**شرح الكود:**

- فى السطر 3 قمنا بالإعلان عن المتغير i الذى يعمل كدليل index فى الدوارة for loop.
- فى السطر 4 نبدأ بناء الدوارة for loop والتى عدد دوراتها 10.
- فى السطر 5 تبدأ الجملة الشرطية select case باختبار أول قيمة من قيم المتغير i (0).
- فتبدأ الدخول على أول جملة من جمل case وأول جملة هى case 0 ومعناها هل قيمة المتغير i تساوى 0 فإذا كانت تساوى 0 يتم تنفيذ السطر 7.

Console.WriteLine("i is Zero");

- ثم يتم الخروج من التركيب select case ويتم العودة مرة أخرى إلى الدوارة for loop فتصبح قيمة i=1 ويتم الدخول على المنشأ Select case

Block ويتم اختبار قيمة i مع الحالة الأولى case 0 فلا يتحقق الشرط لأن قيمة i=1 ثم يتم الدخول على الحالة الثانية case 1 فيتحقق الشرط فيتم تنفيذ السطر 9.

Console.WriteLine("i is one");

- ثم يتم الخروج من المنشأ switch Block الى الدوارة loop for مرة أخرى وتصبح قيمة i=2 ثم يدخل على المنشأ select Block ويحاول اختبار القيمة مع كل حالة وهكذا دواليك حتى تصبح قيمة i أكبر من أو تساوى 10 فيتم الخروج من الدوارة Loop وينتهى عمل البرنامج.

والشكل 1 يوضح نتيجة التنفيذ.

```
C:\Documents and Settings\Administrator\My Docum...
i is Zero
i is one
i is two
i is three
i is four
i is five or more
i is five or more
i is five or more
i is five or more
i is five or more
_
```

شكل 1 تنفيذ التطبيق

## جمل التكرار Looping:

هى مجموعة من الجمل التى تستعمل لتكرار تنفيذ الأوامر أكثر من مرة وهى مهمة جداً ويوجد منها فى لغة VB.NET الجمل التالية:

- جملة for.
- جملة while.
- جملة do...while.

## التكرار باستعمال for Loop:

تستعمل هذه الجملة لتكرار تنفيذ عملية أكثر من مرة وهى أبسط وأشهر أنـــواع جمل التكرار وتأخذ الصورة التالية:

```
For n=val TO x
      Statements
Next n
```

في هذه الصورة تأخذ جملة for الأجزاء التالية:

- for الأمر نفسه المستعمل فى التكرار.
- val القيمة الابتدائية التى يبدأ بها التكرار.
- Statements هى الجملة (أو الجمل) المطلوب تنفيذها داخل التكرار.
- Next هى الجملة المسئولة عن زيادة قيمة n

وسنقوم الآن ببناء برنامج يقوم بحساب الجذر التربيعى لعدد من نطاق 1 إلى 99 وأيضاً نقوم بحساب الخطأ round الذى يحدث عند حساب الجذر التربيعى وذلك باستخدام الدوارة for loop.

## مثال 3: الدوارة for loop:

إذا لم يكن Visual Studio.net يعمل قم بتشغيل Visual Studio.net وذلــك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

أما إذا كان Visual Studio.net يعمل فقم بالذهاب الى قائمة

File▶ Close Solution

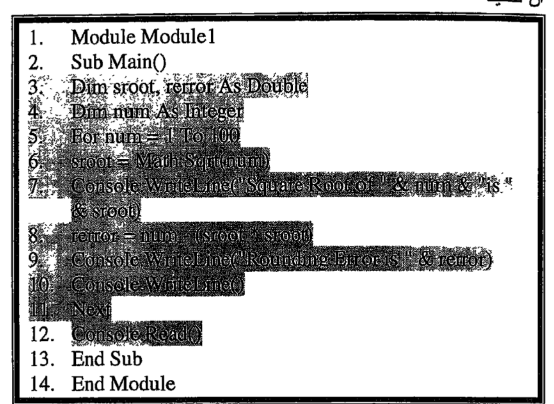
ثم

File▶ New▶ Project

قم بإنشاء create مشروع Project جديد باســتخدام لغــة VB.NET واجعــل

المشروع من نوع Console Application وقم بتسمية المشروع SqrRoot ثم اضغط الزر OK.

عند الضغط على زر Ok تجد أن Visual Studio.net قد قام بكتابة الهيكل الرئيسى للبرنامج وذلك كما يماثل السطور التالية ما عدا المظلل منها الذى لابد أن تضيفه



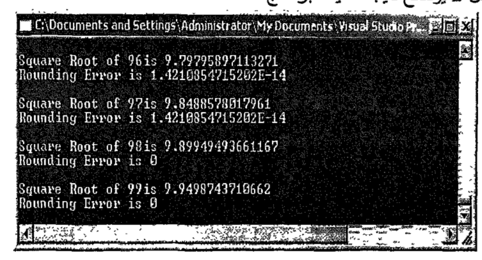
```
1.  Module Module1
2.  Sub Main()
3.  Dim sroot, rerror As Double
4.  Dim num As Integer
5.  For num = 1 To 100
6.  sroot = Math.Sqrt(num)
7.  Console.WriteLine("Square Root of " & num & " is "
    & sroot)
8.  rerror = num - (sroot * sroot)
9.  Console.WriteLine("Rounding Error is " & rerror)
10. Console.WriteLine()
11. Next
12. Console.Read()
13. End Sub
14. End Module
```

## شرح الكود:

- فى السطر 3 تم الإعلان عن متغيرين sroot, rerror من نوع double.
- فى السطر 4 تم الإعلان عن المتغير num وهو من نوع Integer
- فى السطر 5 قمنا ببناء دوارة for loop وعدد دوراتها 100 وذلك لأننا حددنا الشرط num=1 to 100 وجعلنا المتغير num يبدأ بالرقم 1.

- فى السطر 6 قمنا باستدعاء الدالة Sqrt(var) وهى دالة static مبنية فـى الفصيلة Math وهذه الدالة Method تقوم بحساب الجذر التربيعى للمتغير الذى بين القوسين ثم يتم تخزين الناتج فى المتغير sroot.
- فى السطر 7 يتم طباعة قيمة المتغير sroot.
- فى السطر 8 يتم حساب الخطأ Round ويتم تخزين النـاتج فـى المتغيـر rerror.
- فى السطر 9 يتم طباعة قيمة المتغير rerror.

والشكل 2 يوضح نتيجة تنفيذ البرنامج.




شكل 2 تنفيذ التطبيق

## التكرار باستعمال While:

تستخدم جملة While لتكرار العمليات التى تعتمد علـى شـرط معـين وغيـر معروف عدد مرات التكرار وتأخذ الصورة العامة التالية:

```
While (condition)
 Statements
.
.
End While
```

وفى هذه الصورة يتم اختبار الشرط Condition ، فإذا كان صحيحاً فيتم تنفيذ الجمل Statements ويستمر التنفيذ حتى يصبح الشرط غير صحيح.

ولتوضيح ذلك تابع المثال التالي.

## مثال 4: الدوارة while loop:

إذا لم يكن Visual Studio.net يعمل قم بتشغيل Visual Studio.net وذلك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

أما إذا كان Visual Studio.net يعمل فقم بالذهاب الى قائمة

File▶ Close Solution

ثم

File▶ New▶ Project

قم بإنشاء create مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع SqrRoot ثم اضغط الزر OK.

عند الضغط على زر Ok تجد أن Visual Studio.net قد قام بكتابة الهيكل الرئيسى للبرنامج وذلك كما يماثل السطور التالية ما عدا المظلل منها الذى لابد أن تضيفه

```
1.  Module Module1
2.  Sub Main()
3.  Dim i As Integer = 0
4.  Dim MyDate As Date = Now
5.  While (i <= MyDate.DayOfWeek)
6.  Select Case i
7.  Case 0
```

```
8.  Console.WriteLine("DayOfWeek is Sunday")
9.  Case 1
10. Console.WriteLine("DayOfWeek is Monday")
11. Case 2
12. Console.WriteLine("DayOfWeek is Tuesday")
13. Case 3
14. Console.WriteLine("DayOfWeek is Wednesday")
15. Case 4
16. Console.WriteLine("DayOfWeek is Thursday")
17. Case 5
18. Console.WriteLine("DayOfWeek is Friday")
19. Case Else
20. Console.WriteLine("DayOfWeek is Saturday")
21. End Select
22. i += 1
23. End While
24. End Sub
25. End Module
```

**شرح الكود:**

- فى السطر 3 قمنا بالإعلان عن المتغير i من نوع integer وهو المتغير سنقوم باختباره بواسطة الدوارة while.
- فى السطر4 أعلنا عن المتغير MyDate من نوع Date وقمنا بتخصيص الدالة Now (أى أصبحت قيمة المتغير MyDate التاريخ الحالى)
- فى السطر 5 أنشأنا الدوارة While التى تنتهى عند السطر 23 وسنلاحظ أن هذه الدوراة ستظل تعمل طالما كانت قيمة المتغير I أقل من أو تساوى قيمة اليوم الحالى فى الأسبوع وأيام الأسبوع مرتبة من 0~6 حيث أن 0 يعنى الأحد و6 تعنى السبت.

- فى السطر 6 نقوم باختبار قيمة المتغير أى عن طريق البلوك Select case
- فلو كانت قيمة I صفر يتم تنفيذ السطر 8 ثم يتم الخروج من البلوك Select case ويتم تنفيذ السطر22 الذى يزيد قيمة المتغير بواحد فتصبح قيمة i=1 ويتم الرجوع مرة أخرى إلى الدوارة While للتحقق من قيمة I فإذا كان الشرط ما زال صحيحاً يتم الدخول إلى البلوك Select case لاختبار قيمة I وطبع ما يوازيها من اسم اليوم وهكذا دواليك.

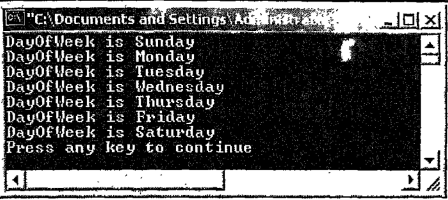
ملحوظة:

- لابد من شحن المتغير الذى سيختبر بواسطة الدوارة while بقيمة ابتدائية Initial Value قبل اختباره بالدوارة while.
- فى السطر 5 نجد بداية بناء الدوارة while وقد وضعنا الشرط (i <= MyDate.DayOfWeek) أى يتم الاستمرار فى الدوران حتى تكون قيمة المتغير أقل من أو تساوى MyDate.DayOfWeek.

ملحوظات:

- لابد من تغيير قيمة المتغير المختبر بالدوارة while داخل المنشأ block التابع للدوارة while وإلا سيتم تنفيذ الدوارة Loop إلي ما لا نهاية.
- فى السطر 21 نجد قوس غلق المنشأ block التابع للدوارة while.

والشكل 3 يوضح نتيجة تنفيذ البرنامج.




شكل 3 تنفيذ التطبيق

## استخدام do --- While:

يستخدم هذا التكرار مثل While ولكن يختلف عنه فى أنه يبدأ أولاً بتنفيذ بعض الجمل ثم يختبر الشرط فى While فى نهاية التركيب فإذا كان صحيحاً ، فيعـاد التكرار مرة أخرى وإلا يتوقف وبالتالى يتم تنفيذ الجمل مرة واحدة على الأقـل حتى لو لم يتحقق الشرط.

وتأخذ الدوارة do --- while الصورة العامة التالية:

```
do
Statements
Loop while (Condition)
```

أو

```
Do
Statements
Loop until val=x
```

فى هذه الصورة يتم تنفيذ الجمل statements أولاً ثم اختبار الشرط الموجود مع While فإذا كان صحيحاً ، فيعود التكرار إلى do ويتم تنفيذ الجمل مرة أخـرى وهكذا.

فى السطور التالية نعرض مثالاً بسيطاً لاستعمال do---While لتحسـين لعبـة

التخمين التى وضحناها فى أمثلة سابقة كما يلى.

## مثال 5: الدوارة do --- while:

إذا لم يكن Visual Studio.net يعمل قم بتشغيل Visual Studio.net وذلك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

أما إذا كان Visual Studio.net يعمل فقم بالذهاب الى قائمة

File▶ Close Solution

ثم

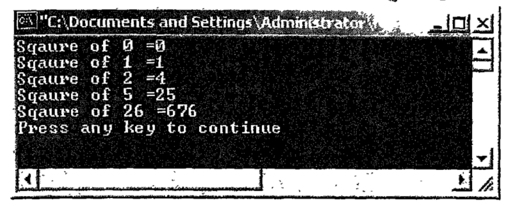
File▶ New▶ Project

قم بإنشاء create مشروع Project جديد باستخدام لغة VB.NET واجعل المشروع من نوع Console Application وقم بتسمية المشروع DWloop ثم اضغط الزر OK.

عند الضغط على زر Ok تجد أن Visual Studio.net قد قام بكتابة الهيكل الرئيسى للبرنامج وذلك كما يماثل السطور التالية ما عدا المظلل منها الذى لابد أن تضيفه.

```
1. Module Module1
2. Sub Main()
3. Dim i As Integer = 0
4. Do
5. Console.Write("Sqaure of " & i & " =")
6. i *= i
7. Console.Write(i & vbCrLf)
8. i += 1
9. Loop While (i < 100)
10. End Sub
11. End Module
```

والشكل التالى يوضح نتيجة التنفيذ.




شكل 4 تنفيذ التطبيق

# الفصل الرابع

## الدوال والمصفوفات

## Methods And Arrays

الدوال والمصفوفات
Methods And Arrays

في هذا الفصل نتعرف علي كيفية استخدام الدوال Methods و المصفوفات Arrays وذلك من خلال النقاط التالية:

1. ماهى الإجراءات Procedures.
2. العناصر المكونة للدالة Method.
3. استدعاء الدوال Method Calling.
4. المصفوفات Arrays.

**مقدمة:**

فى الفصول السابقة تعلمنا استعمال الأنــواع المختلفــة للبيانــات Data Types ورأينا كيف نكرر مهمة معينة باســتخدام جمــل التكــرار Loop Statements ورأينا أيضا كيفية تنفيذ كود بناء على شرط ما وذلك من خلال الجمل الشــرطية Conditional Statements.

وفى هذا الفصل واحدة من أهم الموضوعات فى البرمجة وهو كيفية بناء مقاطع من الكود قابل للاستعمال أكثر من أو ما يسمى بمبدأ reusable code ، ويــتم تحقيق هذا المبدأ عن طريق ما يعرف بالإجراءات Procedures وهــى التــى سنتناولها بالشرح فى هذا الفصل بالإضافة إلي موضوع المصفوفات Arrays.

**تعريف الإجراءات (الدوال) Methods:**

ستلاحظ عزيزى القارئ أنك كلما ازددت تقدماً فى تعلم لغة برمجة كلما وجــدت أن سطور البرنامج أصبحت كثيرة ، وفى كثير من الأحيان تجد نفسك تكرر نفس الكود فى مواضع متعددة من البرنامج وأحياناً فى برامج أخرى. وتسأل نفســك: ألا توجد طريقة سهلة لتكرار هذا الكود!؟ والحــل هــو اســتخدام الإجــراءات Procedures.

والإجراء Procedure هو مقطع (عدد من السطور) يؤدى وظيفة محددة.

وينقسم الإجراء Procedure إلى جزئين هما:

- <u>بناء الإجراء Procedure Implementation:</u> وهى السطور المكونة لهذا الإجراء وجزء البناء يتم ويكتب مرة واحدة. وكل إجراء Procedure يجب أن يكون له إسم واحد فقط وهذا الجزء بمثابة بناء موتور ولكــن لــم يــتم تشغيله.
- <u>استدعاء الإجراء Procedure Calling:</u> وهو نداء الإجراء لأداء عمله فى البرنامج ، أي بمثابة تشغيل الموتور. ونستطيع استدعاء Calling الإجــراء

أى عدد من المرات نريده حسب سير البرنامج ، ويتم استدعاء الإجراء بأن نكتب أسمه فقط.

## أنواع الإجراءات Procedures:

توجد أربعة أنواع من الإجراءات فى لغة VB.net وهم:

- Sub Procedure

وهو أسهل الأنواع الأربعة ولإنشاء إجراء من هذا النوع نبدئه بالكلمة Sub.

- Function procedures

يتميز هذا النوع بقدرته على إرجاع قيمة للبرنامج return a value. ولإنشاء إجراء من هذا النوع نبدئه بالكلمة Function.

- Event-Handling Procedure

هذا النوع من الإجراءات يتم استدعاؤه عند حدوث حدث Event ما مثل الضغط على زر Button أو تحريك الفارة Mouse وسيتم التعرض لهذا النوع فى الفصل العاشر.

- Property Procedure

ويستخدم هذا النوع عند تخصيص assining قيم لخواص أهداف Objects من صنع المبرمج. ولإنشاء إجراء من هذا النوع نبدئه بالكلمة property.

## العمل مع الـ SUB Procedures

الفكرة الرئيسة وراء استخدام أى نوع من أنواع الإجرءات هو تقسيم البرنامج إلى سلسلة من المهام الصغيرة. وكل مهمة task من هذة المهام نقوم بوضعها فى إجراء Procedure أو فصيلة Class. وأنت بتقسيمك البرنامج إلى سلسلة من المهام لها عدد من المزايا هى:

- اختبار كل مهمة task بشكل منفرد.
- تجنب تكرار الكود redundant code.
- تستطيع إنشاء مكتبة من الإجراءات.

- سهولة صيانة البرنامج.

## أين يكتب الاجراء Procedure Location:

من الممكن أن نكتب الإجراء فى أماكن مختلفة مثل:

- تكتب أغلب الإجراءات داخل نطاق ملف الفصيلة Class.
- من الممكن أن نكتب الإجراء داخل وحدة برمجـة Module ووحـدة البرمجة Module ببساطة عبارة عن ملف بة كود VB.net.

وسنرى كلا الطريقتين فى إنشاء الإجراء.

## مثال 1: إنشاء إجراء Procedure:

سنجرب أول طريقة وهى إنشاء إجراء داخل نطاق ملف فصيلة.

إذا لم يكن Visual Studio.net يعمل قم بتشغيل Visual Studio.net وذلـك عن طريق

Start ▶Programs ▶Microsoft Visual Studio.NET▶ Microsoft Visual Studio.NET

أما إذا كان Visual Studio.net يعمل فقم بالذهاب الى قائمة

File▶ Close Solution

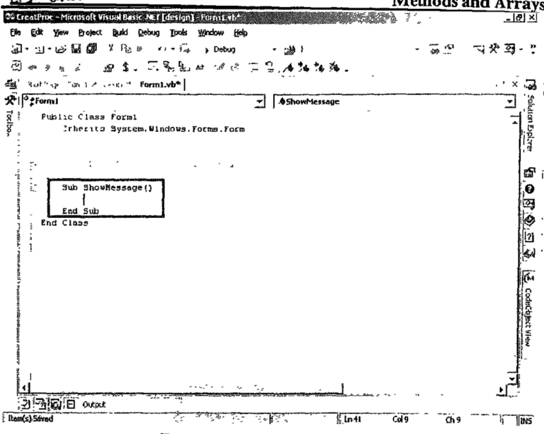
ثم

File▶ New▶ Project

قم بإنشاء create مشروع Project جديد باسـتخدام لغـة VB.NET واجعل المشروع من نوع Windows Application وقم بتسمية المشروع UseProc ثم اضغط الزر OK.

فتجد أنة ظهرت فورم فقم بضـغط الـزر F7 للأنتقـال نافـذة الكـود Code Window التى سنكتب فيها الإجراء وذلك كما فى الشكل 1.

من المفضل كتابة الإجراء الذى ستنشئه قبل نهاية ملف الفصيلة أى قبـل End Class وذلك كما يوضحه الشكل 1.




شكل 1 إنشاء إجراء Procedure

ولإنشاء إجراء Procedure فقم بكتابة ShowMessage Sub ثم اضغط الزر Enter وستجد أنه تم إنشاء الإجراء كما فى شكل 1 بشكل أوتوماتيكى. حتى الآن أنشأنا جسم الإجراء procedure ولكن لم نكتب ما يفعله الإجراء فقم بجعل الإجراء كما فى السطور التالية:

```
Sub ShowMessage()
MessageBox.Show("This IS First Procedure Created")
End Sub
```

هذا الإجراء عند عمله سيؤدى إلى ظهور رسالة "This IS First Procedure Created" ولكن كيف نفعل ذلك؟

لفعل ذلك يجب استدعاء الإجراء Calling Procedure وهو ما سنفعله فى الخطوة القادمة التى من خلالها سنضيف زراً وعند الضغط عليه يستدعى الإجراء.

- قم بالذهاب إلى النموذج Form فى واجهة التصميم (الواجهة التى يكون فيها النموذج Form كما لو كان نافذة صغيرة) أو عن طريق الضغط علـــى زر Shift+F7 ، ثم قم بإضافة زر Button من صندوق الأدوات ToolBox.
- قم بتغيير خاصية النص Text للزر من شاشـــة الخصـــائص Properties Window واجعلها Call Proc.
- قم بالضغط مرتين Double-click على الزر فتجد أنك انتقلت لنافذة الكود لكتابة الحدث click للزر Call Proc:

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Call ShowMessage()
End Sub
```

من خلال هذا أعتقد أنك استنتجت أنه لكى نستدعى إجراء لابد من كتابة الكلمـــة Call متبوعة باسم الإجراء كما يبدو فى السطر المظلل.

قم الآن بترجمة البرنامج عن طريق الضغط على الزر F5 ثـــم اضـــغط الـــزر الموجود في النموذج Form وانظر ماذا ترى!؟

## إنشاء إجراء بمعاملات Procedure with Parameters

هل تتذكر عزيزى القارئ دالة الـــMsgBox() كما فى الشكل التالى:-

```
MsgBox("Hello Sir")
```

ستجد أننا بين الأقواس قد كتبنا بعض الحروف هذه الحروف تسمى معامل الدالة Procedure Parameter وكل معامل لابد أن يكون له نوع ينتمى إليه وســـتجد أن النوع الذى ينتمى إليه هذا المعامل هو String. ومن الممكن أن تكون هنـــاك دوال Methods تأخذ أكثر من معامل مثل الدالــ Format(). ومن هنا يمكننـــا القول بأننا نستطيع أن ننشئ إجراء يستطيع اخذ معـــاملات Parameters وهـــة ماسنفعلة فى الفقرة القادمة.

نريد أن ننشئ إجراء يستقبل معامل من نوع String لكى يتم عرض هذا المعامل فى رسالة وليكن اسم الإجراء الجديد ShowNames.

وذلك كما فى الكود التالى:

قم قبل السطر End Class بكتابة السطور التالية

```
Sub ShowNames(ByVal name As String)
    MessageBox.Show(name)
End Sub
```

ستجد أن هذا الإجراء يشبة الإجراء السابق فيما عدا أننا قد أضفنا name as String (ستجد أن بيئة VS.net قد أضافت المقطع ByVal أوتوماتيكياً) هنا نجد أن اسم المعامل هو name ونوع المعامل String (إنشاء المعامل يشبه الإعلان عن متغير فيما عدا أننا لا نكتب الكلمة Dim) .

بعد ذلك ستجد أننا استدعينا الدالة ()MessageBox.Show وقد مررنا المعامل name أى أنه عند تنفيذ الدالة ()MessageBox سيتم ظهور الاسم الممرر لها فى الرسالة.

## استدعاء إجراء ذو معامل:

بعد أن بنينا الإجراء فى الفقرة السابقة يجئ الدور على كيف سنستعمله أى كيف سنستدعيه.

ولاستدعاء هذا الإجراء قم بإضافة زر جديد للنموذج Form وقم بتغيير خاصية النص Text لتصبح Call Proc With Parm.

قم بالضغط مرتين Double-click على الزر واجعل الكود يشبة الكود التالى:

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Call ShowNames("Kadry")
End Sub
```

ستجد هنا أننا استدعينا الإجراء الجديد كالقديم فيما عدا أننا مررنا passing قيمة حرفية "Kadry" تمثل قيمة المعامل والتى سيتم ظهورها فى رسالة.

قم بترجمة البرنامج ثم قم بالضغط على الزر Call Proc With Parm وأنظر ماذا ترى!؟

## العمل مع Function Procedure:

فى هذه الفقرة سننشئ إجراء من نوع Function وكما قلنا أن الفرق بين الـ Sub Procedure والـ Function Procedure أن الـ Function procedure تقوم بإرجاع قيمة return a value للبرنامج أى أن الدالة مثلا تقوم بعملية حسابية ثم تقوم بإرجاع ناتج هذه العملية الحسابية إلى البرنامج التى يستخدمها داخل البرنامج بشكل ما كما سنرى فى المثال التالى.

سنستعمل الطريقة الثانية فى كتابة إجراء ألا و هى كتابة دخل Module ولذلك –

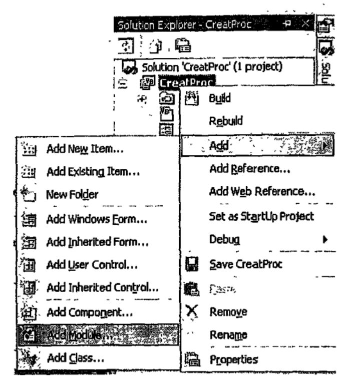
-سنقوم أولا بإضافة Module.

-بعد ذلك نكتب الـ Function procedure داخل هذا الـ Module .

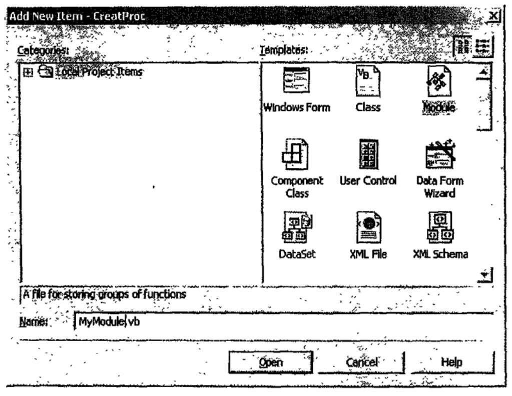
-بعد ذلك نستدعى هذه الدالة من داخل البرنامج.

**إضافة Module**

- قم بالذهاب إلى نافذة مستكشف الحلول Solution Explorer ثم قم بالوقوف على إسم الفصيلة ثم إضغط بالزر الأيمن للماوس ثم أختار ADD ثم Module وذلك كما فى الشكل 2 التالى.
- فيظهر نافذة تطلب منك اسم الـ Module كما فى شكل 3 فاكتب My Module.
- ستجد أن نافذة كود فتحت فاجعل الكود يبدو كما يلى.




شكل 2 إضافة وحدة برمجة Module




شكل 3 إضافة وحدة برمجة Module

```
1.	Module MyModule
2.	Function TriangleArea(ByVal base As Single, ByVal Hight As
	Single) As Single
3.	Dim temp As Single
4.	temp = base * Hight / 2
5.	Return temp
6.	End Function
7.	End Module
```

**شرح الكود:**

- هذة الدالة تقوم بحساب مساحة مثلث (نصف القاعدة مضروب فى الإرتفاع) فتجد فى سطر 2 أننا بدأنا الدالة بكلمة Function وذلك دلالة علـــى أنهـــا ستقوم بإرجاع قيمة. تبع كلمة Function إسم الدالة وهو TriangleArea وهذة الدالة تأخذ معـــاملان مـــن نـــوع Single وهمـــا Base و Hight. (يمثلان قاعدة المثلث وأرتفاعه). تجد بعد ذلك كلمـــة Single AS وهـــى لتحديد نوع البيان المرتجع من الدالة أى أن هذة الدالة لابد وأن ترجع بيـــان من نوع Single .
- فى سطر 3 أعلنا عن متغير temp من نوع temp سنخزن فيـــة مســـاحة المثلث.
- فى السطر نقوم بحساب مساحة المثلث ونقوم بتخزينها فى المتغير temp.
- فى السطر 5 نقوم بإرجاع القيمة عن طريق الأمـــر return وســـتجد أننـــا أرجعنا المتغير temp ولاحظ أنة من نوع Single .
- فى السطر 6 إنهاء الدالة.

## إستدعاء الـFunction Procedure

بعد أن بيننا الدالة TraingleArea() سنقوم باستدعائها

قم بالذهاب إلى النموذج Form فى واجهة التصميم (اضغط Shift+F7) ثم قـــم بإضافة زر جديد وغير خاصية النص Text له إلى Call Triangle.

قم بالضغط علي مرتين Double-click وستجد أنة تم نقلك لنافذة كتابة الكـود فقم بكتابة الكود المظلل التالي:

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button3.Click
1.      Dim Area As Single
2.      Area = TriangleArea(20, 15)
3.      MsgBox(Area)
End Sub
```

## شرح الكود

- فى السطر الأول أعلنا عن متغير من نوع Single لكى أستقبل فية القيمة المرتدة من الدالة TriangleArea() وذلك كما فى السطر 1.
- فى السطر 2 استدعينا الدالة TriangleArea(20,15) وقد مررنا لها معاملان ستتم حساب مساحة المثلث بناء عليهما فتقوم الدالة بحساب مساحة المثلث ثم تقوم بإعادة ناتج العملية ألى البرنامج ويتم تخزينها فى المتغير Area.
- فى السطر 3 نستدعى الدالة MsgBox() ونمرر pass لها المتغير Area كمعامل.
- قم الآن بترجمة البرنامج ثم إضغط على الزر Call Triangle.

ملحوظة:

لا نكتب call عند إستدعاء الإجراء من نوع Function Procedure.

## المصفوفات Arrays:

المصفوفة هى مجموعة من العناصر من نفس النوع يشار إليهم باسم واحد هو اسم المصفوفة.

و المصفوفات نوعان مصفوفة ذات بعد واحد One-dimensional array أو مصفوفة متعددة الأبعاد multi-dimensional Array ، والمصفوفة ذات البعد الواحد هى الأكثر استخداماً.

ونحتاج المصفوفات فى التعامل مع البيانات التى تنتمى الى فئة واحدة فمثلاً عند تمثيل بيانات درجات الطلاب فى مادة الرياضة تمكننا المصفوفات بتمثيلها بسهولة ويسر.

وميزة المصفوفات هى فى طريقة تنظيمها للبيانات بطريقة تجعل معالجتها يسيرة فمثلا عن طريق المصفوفات يمكن ترتيب الطلاب حسب درجاتهم كما يمكن حساب متوسط درجات الطلاب بسهولة .

وتتميز المصفوفات فى لغة VB.NET بأنها أهداف object وهذا يؤدى الى أن المصفوفات الغير مستخدمة يقوم جامع النفايات garbage-collection بإزالتها من الذاكرة memory مما يؤدى الى كفاءة أعلى فى استخدام الذاكرة وسرعة أعلى فى تنفيذ البرنامج عند عمله.

## أنواع المصفوفات:

المصفوفة ذات البعد الواحد One Dimensional Array

للإعلان عن مصفوفة ذات بعد واحد تأخذ الصورة التالية:

```
Dim array-name(size) as type
```

حيث

type هو نوع البيان الذى تمثله المصفوفة (integer,double,...) والقوسان () هما لبيان أننا نعلن عن مصفوفة ذات بعد واحد one dimensional Array.

array-name هو اسم المصفوفة.

size هو عدد العناصر التى تحتويها المصفوفة.

أمثلة على الإعلان عن المصفوفات

```
Dim Degree(30) as Integer
Dim Name(10) as String
Dim Salaries(40) as double
```

ففى المثال

```
Dim Degree(30) as Integer
```

نقوم بإنشاء مصفوفة عدد عناصرها 30 عنصر من نوع integer .

ويتم التعامل مع عناصر المصفوفة برقم يمثل ترتيب هذه العناصر بحيث يكون أول عنصر فى المصفوفة رقمه هو 0 والعنصر الثانى 1 والثالث 2 وهكذا....

فمثلاً لكى نخصص (نخزن assign ) قيمة للعنصر الأول فى المصفوفة degree نقوم بعمل الآتى:

```
Degree(0)=50
```

حيث degree(0) تشير إلى العنصر الأول.

تعال معى نرى مثالاً على الدوال Methods والمصفوفات ذات البعد الواحد.

## مثال 2: استخدام الدوال Methods والمصفوفات Arrays:

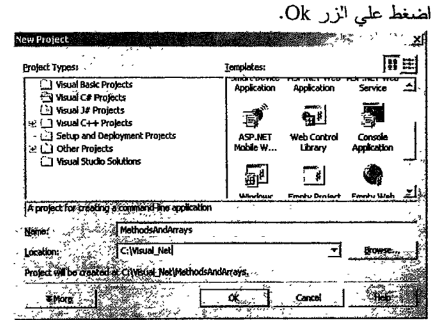
**أولاً: هدف المثال:**

توضيح كيفية إنشاء الدوال Methods وكيفية استدعائها وكيفية إرجاع قيمة من الدالة Method وكيفية استخدام المصفوفات مع الدوال.

**ثانياً: خطوات إنشاء البرنامج:**

1. قم بإنشاء مشروع Project جديد عن طريق القائمة File ثم New ثم Project.
2. اختر نوع المشروع Project Types ليكون Visual VB.NET Projects.
3. اختر قالب المشروع Template ليكون Console Application.
4. اختر اسم المشروع Name ليكون MethodsAndArrays ومكانه

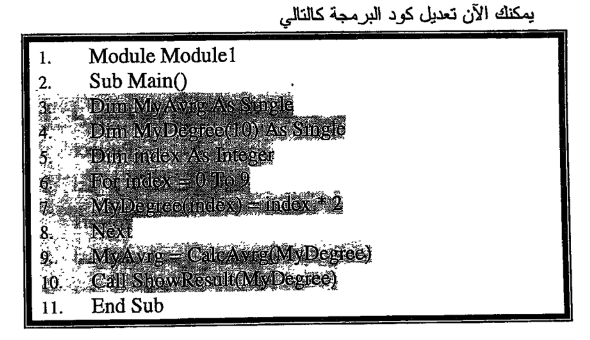
Location ليكون C:\Visual_Net كما هو واضح في شكل 4 ثم اضغط علي الزر Ok.




شكل 4 إنشاء المشروع الخطوة الرابعة

5. سبق لنا شرح كود البرمجة الذي تنشئه لغة VB.NET ويمكنك الرجوع إلي الفصول السابقة لمزيد من الشرح.

يمكنك الآن تعديل كود البرمجة كالتالي



```
1.  Module Module1
2.  Sub Main()
3.  Dim MyAvrg As Single
4.  Dim MyDegree(10) As Single
5.  Dim index As Integer
6.  For index = 0 To 9
7.  MyDegree(index) = index * 2
8.  Next
9.  MyAvrg = CalcAvrg(MyDegree)
10. Call ShowResult(MyDegree)
11. End Sub
```

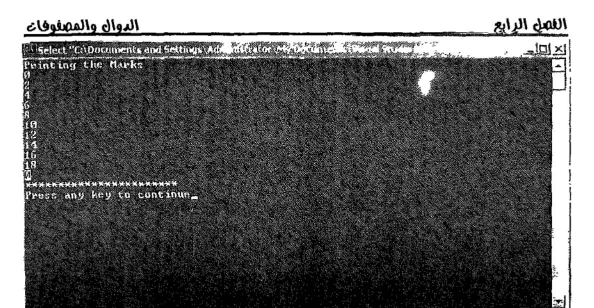
```
12.  Function CalcAvrg(ByVal Degree As Single()) As
     Single
13.  Dim Avrg, Temp, sum As Single
14.  Dim i As Integer
15.  Temp = Degree.Length
16.  For i = 0 To Temp - 1
17.  sum += Degree(i)
18.  Next
19.  Avrg = sum / Temp
20.  Return Avrg
21.  End Function
22.  Sub ShowResult(ByVal Degree As Single())
23.  Dim x, Temp As Integer
24.  Console.WriteLine("Printing the Marks")
25.  Temp = Degree.Length
26.  For x = 0 To Temp - 1
27.  Console.WriteLine(Degree(x))
28.  Next
29.  Console.WriteLine("**********************")
30.  End Sub
31.  End Module
```

## ثالثا: شرح سطور البرمجة:

- في السطر 11 يتم تعريف الدالة Method المسماة () CalcAvrg حيث تقوم هذه الدالة Method بإرجاع قيمة من نوع single وتستقبل معاملاً Parameter واحداً من نوع مصفوفة Array.
- في السطر 13 يتم تعريف ثلاث متغيرات من نوع single. الأول Avrgسنستقبل فية متوسط قيم عناصر المصفوفة. Sum يحتوى مجموع قيم عناصر المصفوفة. الثالث لكى نخزن كم عنصر تحتوى المصفوفة.
- فى السطر 14 أعلنا عن متغير I يعمل كفهرس للدوارة For

- فـــى الســـطر 15 نخـــزن عـــدد عناصـــر المصـــفوفة (عـــن طريـــق Degree.Length) ونخزنها فى المتغير temp
- في السطر 16 يتم إنشاء دوارة loop for بحيث تبدأ هذه الدوارة loop من الصفر حتي أخر عنصر من عناصر المصفوفة Array والذي يتم الحصول عليه عن طريق الخاصية Length ثم يتم جمع عناصـــر هـــذه المصـــفوفة Array ووضع ناتج عملية الجمع في المتغير sum.
- في السطر 19 يتم قسمة مجموع الدرجات علي عـــددهم للحصـــول علـــي متوسط الدرجات.
- في السطر 20 يتم إرجاع قيمة المتغير Avrg عن طريق اســـتخدام كلمـــة return ويتبعها اسم المتغير average مع ملاحظة أن المتغيـــر average من نوع Single مثل نوع القيمة المرتجعة Return Value مـــن الدالـــة CalcAvrg ().
- في السطر 22 يتم تعريف الدالة () ShowResult وهذه الدالة Method لا تقوم بإرجاع أي قيمة وتستقبل معاملاً Parameter مـــن نـــوع مصـــفوفة Array.
- في السطر 24 يتم طباعة رسالة علي شاشة الدوس Dos باستخدام الدالـــة WriteLine() وهي إحدي الدوال سابقة التجهيز Built-in Methods في لغة VB.NET Visual وهذه الدالـــة Method متـــوفرة فـــي الفصـــيلة Console وهي إحدي فصائل classes لغة Visual VB.NET.
- في السطر 26 يتم إنشاء دوارة loop for بحيث تبدأ هذه الدوارة loop من الصفر حتي أخر عنصر من عناصر المصفوفة Array والذي يتم الحصول عليه عن طريق الخاصية Length ثم يتم طباعة عناصر هـــذه المصـــفوفة Array.

- في السطر 35 يتم طباعة نجوم علي شاشة الـ Dos لتبين انتهاء أوامر الدالة Method.
- في السطر 36 نجد End Sub
- في السطر 39 يتم إنشاء الدالة الرئيسية Main().
- في السطر 3 يتم تعريف متغير من نوع single.
- في السطر 4 يتم تعريف متغير من نوع مصفوفة Array من نوع single.
- في السطور من 6 إلي 8 يتم إدخال قيم عناصر المصفوفة Array عن طريق إستعمال الدوارة for.
- في السطر 9 يتم استدعاء Calling الدالة () CalcAvrg مع تمرير المصفوفة MyDegree كمعامل Parameter لها.
- في السطر 10 يتم استدعاء Calling الدالة () ShowResult مع تمرير المصفوفة MyDegree كمعامل Parameter .
- في السطر 11 نجد End Sub والذي يبين نهاية جسم الدالة Method Body الرئيسية Main().
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 5 مع ملاحظة طباعة قيم المتغيرات نتيجة استدعاء الدوال Methods المختلفة في البرنامج.




شكل 5 تنفيذ التطبيق

# الفصل الخامس

## الفصائل والأهداف

## Classes And Objects

في هذا الفصل نتناول مفاهيم البرمجة موجهة الهـدف Object-Oriented Programming (OOP) وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. مفاهيم البرمجة موجهة الهدف Object-Oriented Programming (OOP).
3. الكبسلة Encapsulation.
4. إخفاء البيانات Data Hiding.
5. الوراثة Inheritance.
6. تعدد الأشكال Polymorphism.
7. الفصائل Classes.
8. الأهداف Objects.
9. مكونات الفصيلة Class.
10. أولاً: المتغيرات Variables والدوال Methods.
11. ثانياً: دالة البناء Constructor.

الفصائل والأهداف
Classes And Objects

## مقدمة:

- تطور مفهوم البرمجة عدة مرات منذ بدء ظهور أول لغة برمجة نتيجة ظهور الحاجة إلي إنشاء برامج أكثر تعقيداً وضخامة وبالتالي نجد أن كل مفهوم للبرمجة يقوم بمعالجة بعض المشاكل التي تظهر عند زيادة ضخامة البرنامج الذي تقوم بإنشائه.
- والمقصود بمفهوم البرمجة هو الاستراتيجية العامة وطريقة التفكير لإنشاء البرنامج المطلوب تصميمه.
- ولفهم المقصود بمفهوم البرمجة بطريقة عملية ، تعال معي نتعرف علي مفاهيم البرمجة منذ البداية.

## لغة الأسمبلى (التجميع) Assembly:

- كانت لغة الأسمبلى (التجميع) Assembly من أوائل اللغات التى ظهرت في عالم البرمجة ولكنها كانت لغة صعبة للغاية نتيجة لأنها كانت تعتمد علي كتابة أوامر تتكون من أصفار وآحاد بحيث تكون كل مجموعة من الأصفار والآحاد خاصة بتنفيذ أمر معين ولذلك كان كل من يتعامل معها لابد وأن يكون من الفنيين المحترفين.
- وبعد انتشار استخدام أجهزة الكمبيوتر ، أصبحت هناك حاجة قوية لإنشاء لغات برمجة تستخدم مفردات اللغة بدلاً من الأصفار والآحاد حتي يستطيع الغير فنيين كتابة البرامج التي يحتاجونها دون الدخول في تفاصيل فنية معقدة تحتاج إلي متخصصين.
- وقد كانت هناك عدة محاولات لتحقيق هذا الغرض ولكن معظمها باءت بالفشل نتيجة لعدم تحقيق مبدأ السهولة في التعامل مع اللغة بحيث يستطيع الشخص الغير فني استخدام هذه اللغة.

- وجاء عام 1972 عندما قام دينيس ريتشي Dennies Ritchie بتصميم لغة تستخدم مفردات اللغة الإنجليزية ، ويعتبر استخدامها في غاية السهولة حتي بالنسبة للشخص العادي ، وقد أطلق علي هذه اللغة اسم لغة سي C.
- ومع ظهور لغة سي C لم يصبح العلماء والفنيون فقط هم الـذين يقومـون بالبرمجة ، بل أصبح في استطاعة الكثير من الناس أن يقوموا بتلك العمليـة لأن هذه اللغة تتعامل بمفردات اللغة الإنجليزية المفهومة.
- وأدى تطور لغات البرمجة إلي ظهور الكثير من المفاهيم وكان من أهم تلك المفاهيم هو مفهوم البرمجة التركيبية Structured Programming.

## البرمجة التركيبية Structured Programming:

قبل ظهور البرمجـة التركيبيـة Structured Programming كـان المبرمجون يقومون بكتابة البرنامج كله في ملف واحد بحيث يتم كتابة الأوامـر بطريقة متتابعة في هذا الملف وكانت هذه الطريقة تؤدى إلي الصعوبات التالية:

1. صعوبة كتابة الكود.
2. صعوبة فهم الكود.
3. صعوبة تتبع الأخطاء.
4. عدم القدرة علي تقسيم تنفيذ مهام معينة من البرنامج علي أكثـر مـن شخص لتوفير الوقت والجهد ولسرعة إنجاز البرنامج.
5. عدم القدرة علي إعادة استخدام الكود في برامج أخري.

ولكن البرمجة التركيبية Structured Programming غيرت ذلك المفهوم حيث يقوم المبرمج بتقسيم البرنامج إلي أجزاء (مقاطع) ويعطي لكل جزء اسـماً خاصاً به (وقد تم إطلاق اسم الدوال Functions علي هذه المقاطع) ثم يتم بعـد ذلك استدعاء Calling اسم ذلك الجزء (لتنفيذ مهمة ذلك الجـزء) مـن الدالـة

الرئيسية ()Main بحيث يقوم كل جزء بأداء مهمة محددة ولابد أن تكـون تلـك الدوال Functions مرتبة فى ترتيب منطقى ومترابط.

تقوم فكرة البرمجة التركيبية Structured Programming علي مبدأ بسيط وهو "فرق تسد" حيث أن تقسيم البرنامج إلي أجزاء صغيرة يعني سهولة كتابة كل جزء علي حدة.

وبالتالي نستطيع القول أن البرمجة التركيبية Programming Structured لها المميزات التالية:

1. سهولة كتابة الكود.
2. سهولة فهم الكود.
3. سهولة تتبع الأخطاء وإمكانية عزل الخطأ في جزء معين من البرنامج.
4. القدرة علي تقسيم تنفيذ مهام معينة من البرنامج علي أكثر من شـخص لتوفير الوقت والجهد ولسرعة إنجاز البرنامج.
5. القدرة علي إعادة استخدام الكود في برامج أخري.

ولتوضيح البرمجة التركيبية تعال معـى نأخـذ مثـالاً علـى مبـدأ البرمجـة التركيبية Structured Programming من الواقع العملى وليكن مثلاً حسـاب قيمة الدخل الصافي لمرتبات الموظفين في شركة معينة بعد خصم الضرائب.

قد تكون هذه المهمة صعبة ولكن ما رأيك في تقسيمها بالشكل التالي:

1. السؤال عن اسم الموظف.
2. السؤال عن المرتب الأساسي الموظف.
3. معرفة الشريحة الضريبية الخاصة بالموظف.
4. معرفة النسبة المئوية للضريبة.
5. حساب قيمة الضريبة.
6. حساب الدخل الصافي بعد خصم الضريبة.

7. تكرار نفس العمل لباقي الموظفين.

من الواضح طبعاً سهولة كتابة كل جزء من الأجزاء السابقة وإمكانية توزيع كل مهمة من هذه المهام علي مجموعة من المبرمجين بحيث يقوم كل منهم بتنفيذ الجزء المكلف به وبالتالي يتم توفير وقت وجهد كبيرين ، كما يمكن إعادة استخدام أي جزء من المهام السابقة في برامج أخري ، فمثلاً إذا أردت أن تقوم بتصميم برنامج يقوم بحساب التأمينات للموظفين ، فبالطبع ستحتاج إلي السؤال عن اسم الموظف ومرتبه. كونك سبق لك تنفيذ هذه المهام من قبل ، فيمكنك استخدامها بلا مشاكل في البرنامج الجديد مما يوفر الكثير من الوقت المطلوب لتصميم البرنامج.

إذن نستطيع أن نخرج من المناقشة السابقة بأن البرنامج فى البرمجة التركيبية Structured Programming هو مجموعة من الدوال Functions حيث تتكون الدالة Function من مجموعة من البيانات والأوامر تنفذ جزءاً معيناً من البرنامج.

ثم مع تطور مفاهيم البرمجة وزيادة التعقيد فى البرامج ، كان من اللازم تطوير مفهوم البرمجة التركيبية Structured Programming نتيجة لظهور عيوب في مبدأ البرمجة التركيبية Structured Programming في البرامج الضخمة ، حيث أن البرمجة التركيبية Structured Programming تهتم بتصميم الدوال Functions فقط دون إعطاء أي اهتمام للبيانات نفسها ودون وضع قواعد معينة لكيفية التعامل مع البيانات مما يعتبر قصوراً كبيراً في مفهوم البرمجة التركيبية Structured Programming.

وجاء ذلك التطوير فى شكل جديد وهو البرمجة موجهة الهدف Object Oriented Programming أو اختصاراً OOP وتفيدنا البرمجة موجهة الهدف OOP فى أننا نستطيع تمثيل الأشياء المحيطة بنا تمثيلاً حقيقياً حيث أنه من

الطبيعي أن نفكر في البيانات (مثل اسم الموظف ومرتبه الأساسي) وما يمكننا أن نفعله بهذه البيانات (حساب الضرائب والتأمينات والدخل الصافي) وهذا ما عالجه مفهوم البرمجة موجهة الهدف Object Oriented Programming وهــو مــا سيتضح لنا من خلال النقاط القادمة.

## مفاهيم البرمجة موجهة الأهداف OOP Concepts:

تقوم فكرة البرمجة موجهة الهدف Object Oriented Programming علـــي أربع مبادئ رئيسية هي:

1. الكبسلة Encapsulation.
2. إخفاء البيانات Data Hiding.
3. الوراثة Inheritance.
4. تعدد الأشكال Polymorphism.

سنقوم في السطور القادمة بشرح هذه المبادئ بشئ من الإيجــاز وســنقوم فـــي الفصول القادمة بشرح هذه المبادئ بمزيد من التفصيل وتوضيح كيفية تطبيق هذه المبادئ بلغة Visual Basic.

## أولاً: الكبسلة Encapsulation:

من أحد عيوب البرمجة التركيبيـــة Structured Programming هــو فصـــل البيانات عن الدوال Functions ، ويقوم مفهوم الكبسلة Encapsulation علي ضرورة تجميع البيانات والدوال Functions في عنصر وحيد يحتــوي علـــي جميع البيانات والدوال Functions الخاصة بتنفيذ مهمة معينة ، وهذا العنصـــر يسمي بالفصيلة Class.

نستطيع من مفهوم الكبسلة Encapsulation تحقيق مبدأ إخفاء البيانـــات Data Hiding.

## ثانياً: إخفاء البيانات Data Hiding:

إخفاء البيانات Data Hiding هو كيفية تصميم الفصيلة Class بحيث تبدو مثل الصندوق المقفل للمستخدم بحيث يمكنه التعامل مع هذه الفصيلة Class واستخدامها دون الحاجة لمعرفة الأوامر الموجودة في هذه الفصيلة Class ، فمثلاً أنت تستطيع استخدام التلفاز والتعامل معه دون الحاجة لمعرفة التفاصيل الفنية لكيفية تصنيع التلفاز.

## ثالثاً: الوراثة Inheritance:

يوجد مبدأ عام في البرمجة هو: "لا تقم بإعادة تصميم إطارات السيارة" ، وهذا يعني عدم جدوي تصميم كل شئ من الصفر بل الأفضل هو استخدام الإمكانيات الموجودة لتصميم اختراع جديد حتي لا تقوم بتضييع الكثير من الوقت لاختراع إطار سيارة ثم بعد ذلك تكتشف أن هذا الاختراع موجود من عشرات السنين وربما يكون الاختراع القديم أفضل من اختراعك.

تعتمد فكرة الوراثة Inheritance علي إعادة استخدام الفصائل Classes الموجودة لبناء فصائل Classes جديدة دون الحاجة إلي إعادة كتابة الكود الذي يحتوي علي التفاصيل المتشابهة.

كمثال ، فإن أي موديل جديد للهاتف المحمول يكون له نفس خصائص الموديل القديم (من حيث استقبال وإرسال المكالمات والرسائل القصيرة وتغيير النغمات ... إلخ) مع زيادة تفاصيل جديدة ، وبالطبع فإن الشركة المصنعة للهاتف المحمول لا تعيد تصميم وتنفيذ وظائف الهاتف المحمول بالكامل ولكنها تستخدم الموديل القديم ثم تقوم بالتعديل والتطوير فيه حتي يتم إخراج الشكل الجديد وبالتالي يتم توفير الكثير من الوقت والجهد.

## رابعاً: تعدد الأشكال Polymorphism:

تعتمد فكرة تعدد الأشكال Polymorphism علي تنفيذ الأوامر بطريقة مختلفة علي حسب الموقف ، فمثلاً تجد أن الزر Ok في الهاتف المحمول له أكثر من وظيفة علي حسب الموقف ، بمعني أن هذا الزر يستخدم للرد علي المكالمات عند استقبال المكالمات ، كما يستخدم نفس هذا الزر لقراءة الرسائل القصيرة عند استقبال الرسائل ، كما يستخدم نفس هذا الزر لتغيير النغمات عند وجود الحاجة لتغيير النغمة ... إلخ.

أي أن نفس الزر قام بتنفيذ وظيفة مختلفة علي حسب الموقف وهذا هو مبدأ تعدد الأشكال Polymorphism وهو يقوم بعمل تصميم جيد للبرنامج ، فبدلاً من إنشاء زر للرد علي المكالمات وزر أخر لقراءة الرسائل القصيرة وزر أخر لتغيير النغمات ... إلخ ، فيمكن تصميم زر واحد وإعادة استخدامه وبالتالي نحتاج زراً واحداً فقط لتنفيذ الوظائف المختلفة وإلا كنت ستجد هاتفك المحمول يحتوي علي ما لا يقل عن 200 زر لتنفيذ جميع وظائفه.

وتعتبر عملية إنشاء الفصائل Classes هي المبدأ الأساسي للبرمجة موجهة الهدف Object Oriented Programming حيث يتم تنفيذ الأربعة مبادئ السابق توضيحهم بعد إنشاء الفصائل Classes ولذلك سنقوم في النقطة القادمة بتوضيح كيفية إنشاء الفصائل Classes.

## الفصائل Classes:

إن أى فصيلة Class تتكون من جزئين رئيسيين:

1. الخصائص Attributes وتمثل بالبيانات Data أو المتغيرات Member Variables.

2. السلوك أو الوظيفة Behavior وتمثل بالـدوال Member Functions أو يطلق عليها اسم Methods ، وفـي لغـة Visual Basic يـتم اسـتخدام المصطلح Subroutine غالباً بدلاً من المصطلح Methods.

ولننظر مثلاً إلي الانسان ، فنستطيع القول أن الانسان هو عبارة عـن فصـيلة Class كالتالي:

الانسان له خصائص مثل: الاسم – العمر – الطول – العرض – الوزن – لـون الجلد – لون العينان – الجنس – جائع – شبعان.

الانسان له وظائف مثل: يأكل – يشرب – يبصر – يعـيش – يقـرأ – ينـام – يصحو.

أي أن الفصيلة Class تستخدم لتعريف مجموعة من الخصائص لفئة معينة عـن طريق احتوائها علي مجموعة من البيانات يطلق عليها اسم "متغيـرات العضـو Member Variables" ، كما تتكون من مجموعة من الدوال يطلق عليها اسـم "دوال العضو Member Functions أو Methods أو Subroutines".

لاحظ أنه في تعريف فصيلة الانسان ، فقد ذكرنا أن له اسـماً وعمـراً وطـولاً وعرضاً ... إلخ ، ولكننا لم نحدد قيم هذه البيانات ، ولذلك فإننـا نحتـاج إلـي الأهداف Objects.

## الأهداف Objects:

اتفقنا في النقطة السابقة أن أي انسان له اسم معين فمثلاً يوجد انسان يسمي أحمد وأخر يسمي محمد ... إلخ.

إذن في هذه الحالة ، فإننا نقول أن أحمد هو مثال للانسان لأن لـه اسـم محـدد بالإضافة إلي أن له عمر وطول ووزن ... إلخ ، كما ينطبق نفس هذا الكلام علي محمد وعلي ومصطفي وقدري ... إلخ.

ومن هنا نستطيع تعريف الهدف Object أنه عبارة عن مثال من فصيلة Class معينة بحيث يتم تحديد بيانات محددة لكل هدف Object ومن الطبيعي أن تختلف بيانات كل هدف Object عن الآخر ، فليس من المعقول أن جميع البشر اسمهم أحمد.

أي أن الخصائص العامة لفئة معينة من بيانات ووظائف يتم تحديدها في الفصيلة Class ، أما قيم البيانات فإنه يتم تحديدها في الهدف Object ، وبالتالي لا نحتاج لوجود أكثر من تعريف واحد للفصيلة Class (فمثلاً خصائص الانسان لا تتغير أبداً فكل انسان له اسم وطول ووزن ... إلخ) ولكن سنحتاج عادة لوجود أكثر من هدف Object (بسبب التنوع الكبير في بيانات الاسم والطول والوزن ... إلخ من انسان لأخر).

وفى الحقيقة ، فإن لغة Visual Basic بكاملها هى عبارة عن مجموعة من الفصائل Classes منظمة فى مكتبات (نطاقات) namespaces ، وكل مكتبة (نطاق) namespace تحتوي بداخلها علي مجموعة من الفصائل Classes التي تحدد الخصائص العامة لجزء معين من البرنامج ، وكل ما عليك هو أن تعرف الفصيلة Class التى تؤدى الوظيفة التى تريد عملها فى البرنامج ثم تنشئ منها هدفاً Object ثم تقوم بتحديد قيم بيانات هذا الهدف Object كما يمكنك الاستفادة من وظائف هذا الهدف Object.

## مكونات الفصيلة Class:

تحتوي الفصيلة Class علي عدة أجزاء وسنبدأ بإنشاء فصيلة الانسان لكي نتعرف من خلالها علي هذه المكونات.

## أولاً: المتغيرات Variables والدوال Methods:

نريد الآن أن نقوم بتعريف فصيلة الانسان بحيث أن خصائص الانسان هي أن له اسم وعمر ، كما أن له وظائف هي الجري والسباحة ولعب كرة القدم.

ملحوظة:

لا تقم بكتابة الأوامر التالية ولكن تابع شرحها أولاً حتي تتأكد من فهمك العميق لكل جزء من أجزاء البرنامج ثم سنقوم لاحقاً بتوضيح كيفية تنفيذ هذا البرنامج ولذلك لا تتسرع وتابع الشرح.

```
1:   Class Human
2:         Public Name As String
3:         Public Age As Integer
4:
5:         Public Function Run()
6:            Console.WriteLine("I Run")
7:         End Function
8:
9:         Public Function Swim()
10:           Console.WriteLine("I Swim")
11:        End Function
12:
13:        Public Function PlayFootball()
14:           Console.WriteLine("I Play")
15:        End Function
16:  End Class
```

**والآن إلى تحليل الكود:**

- فى السطر رقم 1 نبدأ بكلمة class وهى الكلمة المستعملة فى إنشاء الفصيلة class (كلمة.class كلمة محجوزة Reserved Word فى اللغة أى مبنية فيها) ثم تلا ذلك اسم الفصيلة class ويمكنك أن تطلق أى اسم تشاء على الفصيلة class بشرط ألا يبدأ برقم ويستحب أن يكون اسماً يدل على عمل الفصيلة class ؛ ومن القواعد المتفق عليها – وليست إجبارية – أن يكون أول حرف فى اسم الفصيلة حرف كبير Capital Letter كما يكون أول حرف فى اسم الدوال الاعضاء Member Function فى الفصيلة حرف كبير Capital Letter.

- في السطرين 2 و 3 قمنا بإنشاء متغيرات الفصيلة class وهذه المتغيرات هي التي تمثل خصائص Attributes الفصيلة class التي نحتاجها وهي الاسم (وهو من نوع String لأنه عبارة عن نص) والعمر (وهو من نوع Integer لأنه عبارة عن رقم).

ملحوظة:

في معظم الأمثلة القادمة ، تجد كلمة Public تسبق اسم المتغير أو الدالة Method وسنقوم بمعرفة معناها في الفصل السادس فتابع شرح الأمثلة ولا تتعجل.

- في السطر رقم 5 بدأنا في إنشاء دالة عضو في الفصيلة Member Method وهي التي تحدد سلوك Behavior الفصيلة Class وهو كما نرى يتكون من ثلاث مقاطع:

الأول: Function وهذا المقطع يبين أننا نقوم بإنشاء دالة Method.

الثاني: Run هو اسم الدالة Method وهذا الاسم لك الحرية في اختياره.

الثالث: ( ) وهذه الأقواس تحدد لنا أن هذه الجملة هي جملة تعريف دالة Method.

- في السطر رقم 6 يتم طباعة رسالة علي شاشة الدوس Dos عن طريق استخدام الفصيلة Console وهي أحد الفصائل classes التي توفرها لنا لغة Visual Basic وهي تحتوي علي الدالة WriteLine() التي تطبع الرسالة التي نريدها علي شاشة الدوس Dos.
- في السطر رقم 7 نجد جملة End Function والتي تبين انتهاء الدالة Run().

- في السطور من 9 إلي 15 قمنا بإنشاء دالتين Methods أخريتين مهمتهما طباعة رسائل مختلفة علي شاشة الدوس Dos.
- في السطر رقم 16 نقوم بإنهاء الفصيلة class المسماة Human عن طريق الجملة End Function.

ملخص المثال:

تعرفنا في هذا المثال علي كيفية إنشاء فصيلة Class تحتوي علي متغيرات Variables ودوال Methods وسنقوم في المثال القادم بتوضيح كيفية إنشاء الفصائل Classes والأهداف Objects في مثال تطبيقي ولكن تأكد أولاً من فهمك للمثال السابق قبل تنفيذ المثال القادم.

## مثال 1: إنشاء الفصائل Classes والأهداف Objects:

أولاً: هدف المثال:

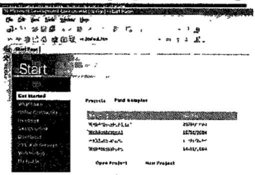
توضيح كيفية إنشاء الفصائل Classes والأهداف Objects وكيفية استخدام البيانات واستدعاء الدوال Method Calling.

ثانياً: خطوات إنشاء البرنامج:

1. افتح البرنامج عن طريق اختيار

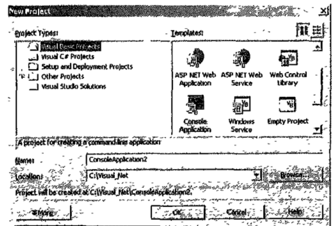
Start → Programs → Microsoft Visual Studio.Net → Microsoft Visual Studio.Net

لتظهر لك نافذة البرنامج كما هو واضح في شكل 1.




شكل 1) الفصائل Classes والأهداف Objects الخطوة الأولي)

2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا الشاشة كما هو واضح في شكل 2.



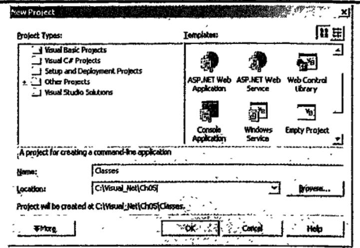

شكل (2 الفصائل Classes والأهداف Objects الخطوة الثانية

3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".

4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".

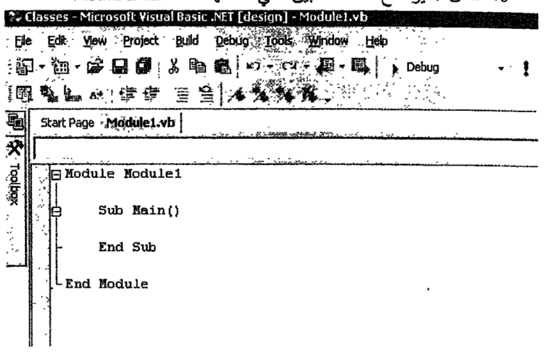
5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Classes وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch05" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 3 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




شكل (3 الفصائل Classes والأهداف Objects الخطوة الخامسة

6. شكل 4 يوضح نافذة التطبيق التي أنشأتها لغة Visual Basic.



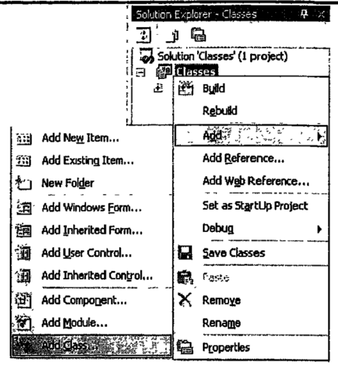

شكل (4 الفصائل Classes والأهداف Objects الخطوة السادسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة حيث سنقوم بشرح الكود المكتوب حتى نستطيع التعديل فيه.

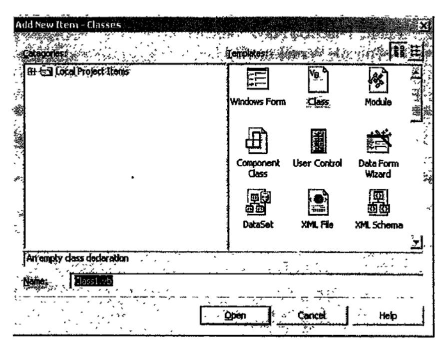
```
1.      Module Module1
2.
3.          Sub Main()
4.
5.          End Sub
6.
7.      End Module
```

- في السطر رقم 1 يتم إنشاء وحدة برمجة Module اسمها Module1.
- في السطر رقم 3 يتم تعريف الدالة الرئيسية Main() وهـــي الدالـــة التـــي يستدعيها نظام التشغيل Operating System عند فتح البرنامج لتنفيذ كود البرمجة بداخلها.
- فى السطر رقم 5 نقوم بإغلاق الدالة الرئيسية Main() عن طريق الجملـــة End Sub.
- فى السطر رقم 7 نقوم بإغلاق وحدة البَرمجة Module عن طريق الجملـــة End Module.

نحتاج الآن لإضافة فصيلة Class جديدة ولذلك قم بالضغط على كلمة Classes (والتي تمثل اسم المشروع Project) من شاشة مستكشف الحـــل Solution Explorer ثم اختر Add ثم Add Class كما هو واضح في شكل 5 ليتم فـــتح الشاشة كما هو واضح في شكل 6.




(شكل 5) إضافة فصيلة Class جديدة




(شكل 6) إضافة فصيلة Class جديدة

يمكنك الآن كتابة اسم للفصيلة Class الجديدة وليكن Human.vb حيث يعتبر امتداد Extension الفصائل Classes في لغة Visual Basic هو (.vb) ثم اضغط الزر Ok ولاحظ إضافة تبويب Tab في أعلي نافذة المشروع Project باسم Human.vb.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Public Class Human
2:      Public Name As String
3:      Public Age As Integer
4:
5:      Public Function Run()
6:          Console.WriteLine("I Run")
7:      End Function
8:
9:      Public Function Swim()
10:         Console.WriteLine("I Swim")
11:     End Function
12:
13:     Public Function PlayFootball()
14:         Console.WriteLine("I Play")
15:     End Function
16: End Class
```

## ثالثاً: تحليل المثال:

- يمكنك الرجوع إلي المثال السابق لمزيد من الشرح لهذه السطور حيث لا يوجد جديد هنا.

نحتاج الآن لكتابة الأوامر في الدالة الرئيسية Main() ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project كما هو واضح في شكل 7.

اضغط هنا

(شكل 7) أوامر الدالة الرئيسية Main()

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:     Sub Main()
4:        Dim h1 As Human = New Human()
5:        h1.Name = "Mostafa"
6:        h1.Age = 26
7:        h1.Run()
8:        h1.Swim()
9:        h1.PlayFootball()
10:
11:
    Console.WriteLine("************************************
    ")
12:
13:       Dim h2 As Human = New Human()
14:       h2.Name = "Kadry"
15:       h2.Age = 30
16:       h2.Run()
17:       h2.Swim()
18:       h2.PlayFootball()
19:
20:
    Console.WriteLine("************************************
    ")
21:    End Sub
22:
23: End Module
```

- في الملف السابق Human.vb قمنا بإنشاء متغيرات ودوال Methods الفصيلة Human ، ولكى نستخدم تلك الفصيلة class يجب أن ننشئ هدفاً object من هذه الفصيلة class ويتم ذلك فى الدالة الرئيسية ()Main وهى الدالة التي يستدعيها نظام التشغيل Operating System لبدء تنفيذ كود البرمجة بداخلها حيث أن الدالة الرئيسية ()Main تأخذ الشكل التالي:

Sub Main ( )

- فى السطر رقم 4 نقوم بإنشاء هدف object ينتمى إلى الفصيلة Human وذلك لكى نستطيع التعامل مع الفصيلة class فنحن لا نستخدم الفصيلة class بل عنصراً منها ، والقاعدة لإنشاء هدف object من أي فصيلة class هى:

Dim objectname As classname = New classname()

كما هو واضح فى السطر رقم 4:

Dim h1 As Human = New Human()

واسم الهدف object قد يكون أى اسم تختاره المهم أن يتبع القاعدة المذكورة سابقاً مع أسماء الفصائل classes ، وكما ذكرنا سابقاً فإن الهدف object المسمي h1 يأخذ خصائص وسلوك الفصيلة Human وهذا ما توضحه السطور من 5 إلى 9.

- ففي السطر رقم 5 نريد تحديد قيمة الخاصية Name للهدف h1 ، ولكى تربط هذه الخاصية بالهدف object المسمي h1 فذلك يتبع القاعدة التالية:

Objectname.variablename = value

كما هو واضح في السطر رقم 5:

h1.Name = "Mostafa"

- وبالمثل فى السطر رقم 6 نكون قد ربطنا الهدف object المسمي h1 بجميع المتغيرات التى قمنا بتحديدها فى الفصيلة Human وهى Name, Age أي الاسم والعمر.

- وأعود وأذكرك عزيزى القارئ أن الهدف object له خصائص وسلوك وقد قمنا بتحديد قيم الخصائص وفيما يلي نقوم باستدعاء calling الدوال Methods الموجودة فى الفصيلة Human وهي الدوال:

Run(), Swim(), PlayFootball().

ولكى نربط الهدف object المسمي h1 بالدالة Method فذلك يتبع القاعدة التالية:

objectname.MethodName ()

كما هو واضح في السطور من 7 إلي 9:

```
h1.Run()
h1.Swim()
h1.PlayFootball()
```

- في السطر رقم 11 نقوم بطباعة نجوم لتمثل فاصلاً بعد طباعة البيانات السابقة.
- في السطور من 13 حتي 20 نقوم بتكرار نفس العمل ولكن مع الهدف h2.
- فى السطر رقم 21 نقوم بإغلاق الدالة الرئيسية ()Main عن طريق الجملة End Sub.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 8 مع ملاحظة طباعة قيم متغيرات الفصيلة Class نتيجة استدعاء الدوال Methods السابق شرحها.

```
C:\Visual_Net\Ch05\Classes\bin\Classes.exe
I Run
I Swim
I Play
************************************
I Run
I Swim
I Play
************************************
Press any key to continue_
```

شكل (8) تنفيذ البرنامج

يمكنك إغلاق البرنامج الآن عن طريق فتح قائمة File ثـم Close Solution استعداداً للمثال التالي.

## ثانياً: دالة البناء Constructor:

كما تعلمنا في الفصول السابقة ، فإنه يمكننا إنشاء متغير وتحديد قيمـة ابتدائيـة Initial Value له بالشكل التالي:

Dim Age As Integer = 25

ويمكننا تغيير القيمة الابتدائية Initial Value للمتغير في أي سطر من سـطور البرمجة حيث أن القيمة الابتدائية Initial Value تضمن فقط أن المتغير له قيمة محددة دائماً ولكن لا يوجد ما يمنع تغييرها فيما بعد.

السؤال الآن: كيف نحدد قيمة ابتدائية Initial Value لمتغيرات الفصيلة class؟

الإجابة: تتوفر في الفصيلة class دالة خاصة تسمي دالة البنـاء constructor وهي المسئولة عن إعطاء قيمة ابتدائية Initial Value لمتغيرات الفصيلة class حيث يتم استدعاء دالة البناء constructor تلقائياً عند إنشاء هدف object مـن الفصيلة class ، ويمكن لدالة البناء constructor أن تسـتقبل أي عـدد مـن المعاملات parameters لتحديد القيم الابتدائية Initial Value للمتغيرات فـي الفصيلة class.

تختلف دالة البناء constructor عن أي دالة عادية في الآتي:

1. لا بد أن يكون اسم دالة البناء constructor هو New.
2. لا تقوم دالة البناء constructor بإرجاع أي قيمة.

فمثلاً ، إذا قمنا بإنشاء فصيلة class باسم Human فإن الدوال الآتية تعتبر مثالاً صحيحاً لدالة البناء constructor:

```
Sub New ()
Sub New (Name As String)
```

Sub New (Age As Integer)
Sub New (n As String, a As Integer)

أما الدوال الآتية فتعتبر مثالاً غير صحيح لدالة البناء constructor:

Sub Human ()
Sub New (Name As String) As String
Sub New () As Integer

حيث أن الدالة الأولي لها اسم مختلف عن الاسم New وأما الدالتان الثانية والثالثة فتقومان بإرجاع قيمة بصرف النظر عن نوع القيمة المرتجعة Return Value.

ملحوظة:

إذا أخطأت في كتابة دالة البناء constructor فذلك يعني أن ما قمت بتعريفه هو دالة Method عادية وبالتالي لن يتم استدعاؤها عند إنشاء هدف object من الفصيلة class وبالتالي لن يمكنك إعطاء قيم ابتدائية Initial Values لمتغيرات الفصيلة class ، فكن حذراً في كتابة اسم دالة البناء constructor بطريقة سليمة وإلا ستحصل علي نتائج لا ترغب فيها.

المثال التالي يوضح كيفية استخدام دالة البناء constructor.

## مثال 2: دالة البناء constructor:

أولاً: هدف المثال:

توضيح كيفية استخدام دالة البناء constructor لإعطاء قيم ابتدائية Initial Values لمتغيرات الفصيلة class.

ثانياً: خطوات إنشاء البرنامج:

1. تأكد أولاً من فتح البرنامج.
2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم

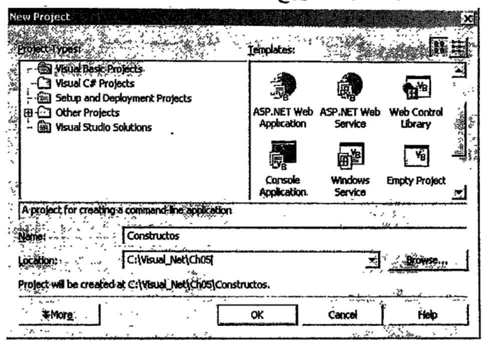
New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا شاشة إنشاء مشروع جديد New Project.

3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".

4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".

5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Constructors وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch05" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 9 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




(شكل 9) دالة البناء Constructor الخطوة الخامسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبـــرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي المثال السابق لمزيد من الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Human كما شرحنا فـــي المثـــال السابق ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:  Public Class Human
2:      Public Name As String
3:      Public Age As Integer
4:
5:      Public Function Run()
6:          Console.WriteLine("I Run")
7:      End Function
8:
9:      Public Function Swim()
10:         Console.WriteLine("I Swim")
11:     End Function
12:
13:     Public Function PlayFootball()
14:         Console.WriteLine("I Play")
15:     End Function
16:
17:     Sub New(ByVal n As String, ByVal a As Integer)
18:         Name = n
19:         Age = a
20:     End Sub
21:
22:     Public Function DisplayDetails()
23:         Console.WriteLine("Name: " & Name)
24:         Console.WriteLine("Age: " & Age & " Years Old")
25:     End Function
26: End Class
```

## ثالثاً: تحليل المثال:

- في السطر رقم 1 يتم إنشاء فصيلة class جديدة باسم Human.

- في السطرين 2 و 3 يتم تعريف عضوين Members للفصــيلة Human حيث يمثل العضو الأول اسم الانسان ويمثل العضو الثاني عمر الانسان.
- في السطور من 5 إلي 15 يتم تعريــف جميــع دوال Methods الفصــيلة Human ولا جديد هنا.
- في السطر رقم 17 يتم تعريف دالة البناء constructor للفصيلة Human حيث تجد أن اسم الدالة لها الاسم New ولا تقوم الدالة بإرجاع أي قيمــة. تستقبل دالة البناء constructor معاملين Parameters ويتم استخدام هذين المعاملين Parameters لتحديد القيمة الابتدائية Initial Value لمتغيــرات الفصيلة Human كما هو واضح في السطرين 18 و 19 مع ملاحظــة أن المتغير Name المستخدم في السطر رقم 18 هو عضو من أعضاء الفصيلة Human كما تم تعريفه في السطر 2 وبالمثل المتغيــر Age والــذي تــم تعريفه في السطر 3.
- في السطر رقم 22 يتم تعريف الدالــة DisplayDetails() وهــذه الدالــة Method مسئولة عن طباعة قيم متغيرات الفصــيلة Human كمــا هــو واضح في السطرين 23 و 24.

نحتاج الآن لكتابة الأوامر في الدالة الرئيسية Main() ويتم ذلــك عــن طريــق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافــذة المشــروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:   Module Module1
2:
3:       Sub Main()
4:           Dim h1 As Human = New Human("Mostafa", 26)
5:           h1.Run()
```

```
6:          h1.Swim()
7:          h1.PlayFootball()
8:          h1.DisplayDetails()
9:
10:
     Console.WriteLine("**********************************
     ")
11:
12:         Dim h2 As Human = New Human("Kadry", 30)
13:         h2.Run()
14:         h2.Swim()
15:         h2.PlayFootball()
16:         h2.DisplayDetails()
17:
18:
     Console.WriteLine("**********************************
     ")
19:     End Sub
20:
21: End Module
```

- في السطر رقم 3 يتم إنشاء الدالة الرئيسية Main().
- في السطر رقم 4 يتم إنشاء هدف object من نوع الفصيلة Human مع تمرير المعاملات Parameters لدالة البناء constructor وهذه القيم تستخدم لإعطاء قيمة ابتدائية Initial Value لمتغيرات الفصيلة Human. لاحظ أن المعامل Parameter الأول من نوع String والمعامل Parameter الثاني من نوع Integer كما حددناه في دالة البناء constructor في الملف السابق.
- في السطور من 5 إلي 7 يتم استدعاء دوال Methods الهدف object المسمي h1 بالطريقة العادية حيث أن وجود دالة بناء constructor لا يمنع استدعاء الدوال Methods.

- في السطر رقم 8 يتم استدعاء الدالة DisplayDetails() والتي تم تعريفها في الملف السابق وتقوم بطباعة قيم متغيرات الفصيلة Human.
- في السطر رقم 10 نقوم بطباعة نجوم لتمثل فاصلاً بعد طباعة البيانات السابقة.
- في السطور من 12 حتى 18 نقوم بتكرار نفس العمل ولكن مع الهدف h2.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 10 مع ملاحظة طباعة قيم متغيرات الفصيلة نتيجة استدعاء الدالة DisplayDetails().

```
C:\Visual_Net\Ch05\Constructors\bin\Constructors.exe
I Run
I Swim
I Play
Name: Mostafa
Age: 26 Years Old
****************************************
I Run
I Swim
I Play
Name: Kadry
Age: 30 Years Old
****************************************
Press any key to continue_
```

(شكل 10) دالة البناء constructor

# الفصل السادس

## الكبسلة والوراثة والفصائل النهائية والمعدلات

## Encapsulation, Inheritance, Sealed Classes And Modifiers




في هذا الفصل نستكمل شرح مفاهيم البرمجة موجهة الهدف Object-Oriented Programming (OOP) حيث نتعرف على مبدأ الكبسلة Encapsulation ومبدأ الوراثة Inheritance والفصائل النهائية Sealed Classes واستخدام المعدلات Modifiers وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. الكبسلة Encapsulation.
3. الوراثة Inheritance.
4. فائدة الوراثة Inheritance.
5. الصيغة العامة للوراثة General Syntax for Inheritance.
6. الفصائل النهائية Sealed Classes.
7. المعدلات Modifiers.
8. المعدل العام Public.
9. المعدل الخاص Private.
10. المعدل المحمي Protected.
11. المعدل الصديق Friend.

**مقدمة:**

- تعرفنا في الفصل السابق علي بعض من مبادئ البرمجة موجهة الهدف OOP ونستكمل في هذا الفصل شرح المزيد من هذه المبادئ.
- سوف نتعرف في هذا الفصل علي مبدأ الكبسلة Encapsulation والوراثة Inheritance والفصائل النهائية Sealed Classes واستخدام المعدلات Modifiers ، وسنبدأ بالتعرف علي مبدأ الكبسلة Encapsulation.

## الكبسلة Encapsulation:

- يقوم مفهوم الكبسلة Encapsulation علي ضرورة تجميع البيانات والدوال Functions في عنصر وحيد يسمي بالفصيلة Class وهو يحتوي علي جميع البيانات والدوال Functions الخاصة بتنفيذ مهمة معينة.
- وقد تعرفنا في الفصل السابق علي كيفية إنشاء الفصيلة Class كما تعرفنا علي أهم مكونات الفصيلة Class ، وبذلك نكون قد نفذنا مبدأ الكبسلة Encapsulation بمجرد إنشائنا لأي فصيلة Class ، ولكن تتبقي نقطة صغيرة ، فإن تحقيق مبدأ الكبسلة Encapsulation كاملاً لا يتأتي إلا باستخدام المعدلات Modifiers والتي سنتعرف عليها لاحقاً في هذا الفصل.
- قبل البدء في موضوع المعدلات Modifiers ، فلا بد أولاً أن نتعرف علي مبدأ الوراثة Inheritance.

## الوراثة Inheritance:

- تعتمد فكرة الوراثة Inheritance علي إعادة استخدام الفصائل Classes الموجودة لبناء فصائل Classes جديدة دون الحاجة إلي إعادة كتابة الكود الذي يحتوي علي التفاصيل المتشابهة.
- ولتوضيح هذا المبدأ ، تعال نفترض وجود فصيلة class للمهندس بحيث تحتوي علي المتغيرات التالية: الاسم والمدينة والعمر والجامعة والقسم

والشعبة ، كما سنفترض وجود فصيلة class للمحاسبين بحيث تحتوي على المتغيرات التالية: الاسم والمدينة والعمر والجامعة وعدد سنوات الخبرة.

- وفي كلا الفصيلتين classes نريد دالة Method لطباعة بيانات كلا الفصيلتين classes وليكن اسمها هو PrintData( ).
- إذا نظرت لتكوين الفصائل classes في النقطتين السابقتين ، فستجد أن المتغيرات الخاصة بالاسم والمدينة والعمر والجامعة هي متغيرات متشابهة وأيضاً الدالة PrintData( ) هي دالة Method مشتركة.
- ونتيجة لاحتياجنا إلى تقليل الكود المكتوب ، إذن فمن الأفضل أن نقوم بإنشاء فصيلة class تحتوى على المتغيرات والدوال Methods المشتركة ثم نقوم بالوراثة Inheritance من هذه الفصيلة class.
- لاحظ أن غرض الوراثة Inheritance الأساسي هو تقليل تكرار المتغيرات والدوال Methods من أجل توفير تكرار الأوامر ولذلك فإننا سنقوم بإنشاء فصيلة class تحتوى على البيانات والدوال Methods المتشابهة وسنطلق عليها اسم Employee ، وبذلك سنعيد بناء الفصائل classes لتصبح كالآتي:
- الفصيلة Employee ستحتوي على المتغيرات التالية: الاسم والمدينة والعمر والجامعة (أي البيانات المتشابهة) كما تحتوي على الدالة PrintData( ).
- الفصيلة Engineer ستحتوي على المتغيرات التالية: القسم والشعبة كما تحتوي على الدالة PrintData( ).
- الفصيلة Accountant ستحتوي على المتغيرات التالية: عدد سنوات الخبرة كما تحتوي على الدالة PrintData( ).

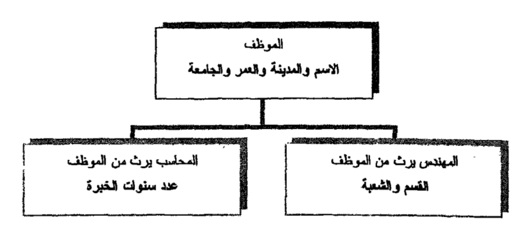
- وفي هذه الحالة ، نستطيع أن نقول أن الفصيلة Engineer و Accountant ترثان من الفصيلة Employee.
- ولتوضيح مبدأ الوراثة Inheritance بشكل علمي ، فلنفترض وجود فصيلة class تسمي B ترث من فصيلة class تسمي A ، إذن فإن الفصيلة B تكون لها جميع صفات الفصيلة A وتزيد عليها ببعض الصفات الأخرى.
- فمثلاً المهندس يرث صفاته من الموظف ويزيد عليها ببعض الصفات الأخرى التي تميزه.
- وفي هذه الحالة نجد أن الفصيلة A تسمى الفصيلة الأم Parent class أو الفصيلة الأساسية Base class أو الفصيلة العليا Super class وكلها مسميات مختلفة لنفس المصطلح ولكن التسمية الغالبة في لغة Visual Basic هي الفصيلة الأساسية Base class.
- أما الفصيلة B فتسمى الفصيلة الصغرى Child class أو الفصيلة المشتقة Derived class أو الفصيلة الفرعية Sub class وكلها مسميات مختلفة لنفس المصطلح ولكن التسمية الغالبة في لغة Visual Basic هي الفصيلة الصغرى Child class.

## فائدة الوراثة Inheritance:

- تمنع الوراثة Inheritance تكرار البيانات المتشابهة أكثر من مرة فنجد أنه بدلاً من تكرار صفات المهندس والمحاسب بالشكل التالي:

| المهندس | المحاسب |
|---|---|
| الاسم والمدينة والعمر والجامعة والقسم والشعبة. | الاسم والمدينة والعمر والجامعة وعدد سنوات الخبرة. |

■ فإننا نعيد الوصف بالشكل التالي:




■ يتضح من الشكل السابق أننا منعنا تكرار البيانات المتشابهة وقمنا بتصميم شكل هرمي Hierarchy يوضح الصفات المشتركة والصفات المميزة لكل فصيلة class.

## الصيغة العامة للوراثة General Syntax for Inheritance:

تتم الوراثة Inheritance في البرمجة بكتابة الكود بالشكل العام التالي:

```
class <Child Class>
Inherits <Base Class>
```

أي أنه يتم كتابة اسم الفصيلة الصغرى Child class ثم يجب النزول إلى سطر جديد (وإلا سيحدث خطأ في ترجمة البرنامج Compile) ثم كتابة كلمة Inherits ثم اسم الفصيلة الأساسية Base Class ، فمثلاً لإنشاء فصيلة class باسم Engineer ترث من فصيلة class تسمى Employee فإننا نكتب الكود التالي:

```
public class Engineer
Inherits Employee
        //Engineer class attributes and methods
End Class
```

مع ملاحظة أن الفصيلة Employee لا بد وأن يكون قد سبق تعريفها وتحديد خصائصها.

المثال التالي يوضح كيفية تنفيذ مبدأ الوراثة Inheritance في البرمجة.

## مثال 1: الوراثة Inheritance:

أولاً: هدف المثال:

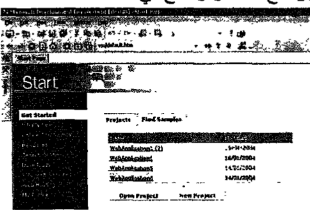
توضيح كيفية تنفيذ مبدأ الوراثة Inheritance في البرمجة.

ثانياً: خطوات إنشاء البرنامج:

1. افتح البرنامج عن طريق اختيار

Start → Programs → Microsoft Visual Studio.Net → Microsoft Visual Studio.Net

لتظهر لك نافذة البرنامج كما هو واضح في شكل 1.




(شكل 1) الوراثة Inheritance الخطوة الأولي

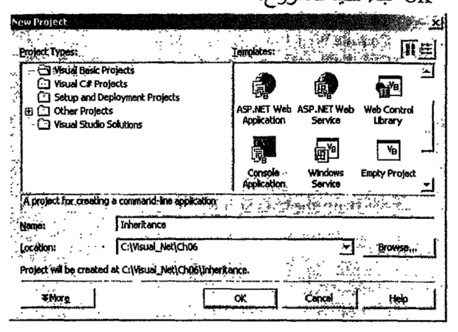
2. لإنشاء مشروع Project جديد فيمكنك الضغط علي الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا شاشة إنشاء مشروع جديد New Project.

3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".

4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".

5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Inheritance وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch06" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 2 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




شكل 2) الوراثة Inheritance الخطوة الخامسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السابقة لمزيد من الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Employee كما شرحنا في الفصول السابقة ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:  Public Class Employee
2:     Public name As String
3:     Public city As String
4:     Public age As Byte
5:     Public university As String
6:
7:     Public Function PrintData()
8:        Console.WriteLine("Name is " & name)
9:        Console.WriteLine("City is " & city)
10:       Console.WriteLine("Age is " & age)
11:       Console.WriteLine("University is " & university)
12:    End Function
13:
14: End Class
15:
16: Public Class Engineer
17:    Inherits Employee
18:    Public department As String
19:    Public section As String
20:
21:    Public Function PrintEngineerData()
22:       MyBase.PrintData()
23:       Console.WriteLine("Department is " + department)
24:       Console.WriteLine("Section is " + section)
25:    End Function
26: End Class
27:
28: Public Class Accountant
29:    Inherits Employee
30:    Public years As Byte
31:
```

```
32:     Public Function PrintAccountantData()
33:         MyBase.PrintData()
34:         Console.WriteLine("Years of experience is " & years)
35:     End Function
36: End Class
```

## ثالثاً: تحليل المثال:

- في السطور من 1 إلي 14 يتم تعريف الفصيلة Employee ولا جديد هنا.
- في السطر رقم 16 يتم إنشاء الفصيلة Engineer التي ترث من الفصيلة Employee.
- في السطور من 18 إلي 26 يتم تعريف الفصيلة Engineer ولا جديد هنا سوي في السطر رقم 22 حيث يتم استخدام الكلمة المحجوزة MyBase لاستدعاء الدالة PrintEngineerData( ) حيث تستخدم هذه الكلمة إذا أردنا استدعاء أي عضو من أعضاء الفصيلة الأساسية Base Class ، وقد استخدمناها هنا حتي نستفيد من البيانات السابق طباعتها بدلاً من إعادة كتابة كود طباعة هذه البيانات مرة أخري.
- في السطور من 28 إلي 36 يتم تكرار نفس الشئ ولكن مع الفصيلة Accountant ولا جديد هنا.

نحتاج الآن لكتابة الأوامر في الدالة الرئيسية Main() ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:      Sub Main()
4:          Dim e As Engineer = New Engineer()
```

```
5:        e.name = "Mostafa"
6:        e.city = "Alex"
7:        e.age = 26
8:        e.university = "Alex"
9:        e.department = "Electrical"
10:       e.section = "Communications"
11:       e.PrintEngineerData()
12:
13:       Console.WriteLine("**************************")
14:
15:       Dim a As Accountant = New Accountant()
16:       a.name = "Ashraf"
17:       a.city = "Cairo"
18:       a.age = 30
19:       a.university = "Cairo"
20:       a.years = 7
21:       a.PrintAccountantData()
22:    End Sub
23:
24: End Module
```

- فى السطر رقم 4 نقوم بإنشاء هدف object من نوع الفصيلة Engineer.
- في السطور من 5 إلي 11 نقوم باستدعاء متغيرات ودوال Methods الهدف e.
- في السطر رقم 13 نقوم بطباعة نجوم لتمثل فاصلاً بعد طباعة البيانات السابقة.
- في السطور من 15 حتي 21 نقوم بتكرار نفس العمل ولكن مع الهدف a.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 3.

```
C:\Visual_Net\Ch06\Inheritance\bin\Inheritance.exe
Name is Mostafa
City is Alex
Age is 26
University is Alex
Department is Electrical
Section is Communications
**************************
Name is Ashraf
City is Cairo
Age is 30
University is Cairo
Years of experience is 7
Press any key to continue_
```

(شكل 3) تنفيذ البرنامج

ملحوظات:

- تم فصل المتغيرات والدوال Methods المتشابهة في فصيلة class منفصلة وعند الحاجة إلي استخدام هذه المتغيرات والدوال Methods في فصيلة class جديدة ، فإننا نرث من الفصيلة الأساسية Base Class.
- علي الرغم من أن المتغير name غير موجود مباشرة في الفصيلة Engineer ، إلا أنه أمكننا استدعاؤه كما هو واضح في السطر رقم 5 حيث يظهر هنا مبدأ الوراثة Inheritance ، فحيث أن الفصيلة Engineer ترث من الفصيلة Employee ، فذلك يعني أنها ترث جميع الخواص من الفصيلة الأساسية Base class بما فيها المتغير name ولذلك أمكننا استدعاؤه بلا مشاكل.
- يتضح لنا أن استخدام مبدأ الوراثة Inheritance يعطي تصنيفاً أفضل للفصائل classes ويجنبنا تكرار المتغيرات والدوال Methods المتشابهة مما يوفر الوقت والجهد.

## الفصائل النهائية Sealed Classes:

الفصائل النهائية Sealed classes هي الفصائل classes التي لا يمكنك الوراثة Inheritance منها ولكن يمكنك إنشاء هدف object منها فقط ، ويتم استخدام الفصائل النهائية Sealed classes في حالة إنشائنا لفصيلة Class لا نريد

إضافة أي أعضاء لها (سواء متغيرات أو دوال Methods) حيث أن الوراثة Inheritance تقوم بإضافة أعضاء أخري للفصيلة الأساسية Base Class كما تعلمنا في النقطة السابقة.

ويكون شكل الإعلان عن الفصائل النهائية Sealed classes كالتالي:

```
Public NotInheritable class Person
        //methods and attributes for class Person
End Class
```

المثال التالي يوضح استخدام الفصائل النهائية Sealed classes.

## مثال 2: الفصائل النهائية Sealed classes:

أولاً: هدف المثال:

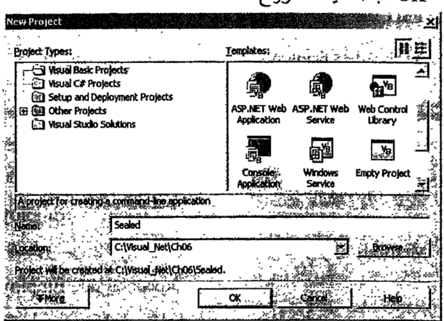
توضيح كيفية استخدام الفصائل النهائية Sealed classes.

ثانياً: خطوات إنشاء البرنامج:

1. تأكد أولاً من فتح البرنامج.
2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا شاشة إنشاء مشروع جديد New Project.
3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".
4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".

5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Sealed وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch06" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 4 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




(شكل 4) الفصائل النهائية Sealed classes الخطوة الخامسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السابقة لمزيد من الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Employee كما شرحنا في الفصول السابقة ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:  Public NotInheritable Class Employee
2:      Public name As String
3:      Public city As String
4:      Public age As Byte
5:      Public university As String
```

```
6:
7:      Public Function PrintData()
8:          Console.WriteLine("Name is " & name)
9:          Console.WriteLine("City is " & city)
10:         Console.WriteLine("Age is " & age)
11:         Console.WriteLine("University is " & university)
12:     End Function
13:
14: End Class
15:
16: 'Public Class Engineer
17: 'Inherits Employee
18: 'End Class
```

## ثالثاً: تحليل المثال:

- في السطر رقم 1 يتم تعريف فصيلة نهائية Sealed class باسم Employee.
- في السطور من 2 إلي 14 يتم تعريف متغيرات ودوال Methods الفصيلة Employee بالطريقة العادية.
- في السطور من 16 إلي 18 يتم توضيح خطأ شائع حيث يتم إنشاء فصيلة class باسم Engineer وهي ترث من الفصيلة Employee وهذا يسبب خطأ في الترجمة Compile لأن الفصيلة النهائية Sealed class لا يمكن الوراثة Inheritance منها ولذلك تم وضع تعليقات Comments علي هذه السطور حتي يمكن تنفيذ البرنامج.

نحتاج الآن لكتابة الأوامر في الدالة الرئيسية ()Main ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:     Sub Main()
4:        Dim e As Employee = New Employee()
5:        e.name = "Mostafa"
6:        e.city = "Alex"
7:        e.age = 26
8:        e.university = "Alex"
9:        e.PrintData()
10:
11:    End Sub
12:
13: End Module
```

- في السطر رقم 4 يتم إنشاء هدف object من نوع الفصيلة Employee وكما ذكرنا ، فإن إنشاء هدف object من فصيلة نهائية Sealed class مسموح به ولا يسبب أي خطأ.
- في السطور من 5 إلي 9 يتم استدعاء متغيرات ودوال Methods الفصيلة Employee بالطريقة العادية.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 5.

```
C:\Visual_Net\Ch06\Sealed\bin\Sealed.exe
Name is Mostafa
City is Alex
Age is 26
University is Alex
Press any key to continue
```

(شكل 5) الفصائل النهائية Sealed classes

## المعدلات Modifiers:

لنفترض أننا قمنا بإنشاء فصيلة Class تحتوي علي متغير اسمه Age ليمثل عمر موظف ما ، وقام أحد المبرمجين بكتابة الكود التالي:

```
1: Public class Employee
2:         Public Age As Integer
3: End Class
```

ثم تم كتابة الأوامر التالية في الدالة الرئيسية Main()

```
1: Sub Main()
2:         Dim e As Employee = New Employee()
3:         e.Age = -30
4: End Sub
```

إذا نظرنا للكود السابق ، فنجد أننا أنشأنا هدفاً object من نوع Employee ثم قمنا بتحديد قيمة المتغير Age لتكون 30-.

إلي هنا لا نجد أي خطأ في البرنامج من وجهة نظر المترجم Complier ولكن من وجهة نظر الانسان ، فهذا البرنامج يحتوي علي خطأ كبير لأن العمر لا يمكن أن يكون سالباً بأي حال من الأحوال.

إذن ما الحل إذا أردنا أن نقوم بمنع المستخدم من إدخال قيمة سالبة للعمر؟

الحل يكمن في استخدام المعدلات Modifiers.

## تعريف المعدلات Modifiers:

إن المعدل Modifier هو كلمة من ضمن عدة كلمات محجوزة (Reserved keywords) في اللغة تستخدم فى تحديد التوصل Access إلى الفصيلة Class ودوالها Methods ومتغيراتها ، ومن ضمن هذه المعدلات Modifiers:

Public, Protected, Private, Friend, Protected Friend.

وباستخدام هذه المعدلات Modifiers يستطيع المبرمج أن يحدد أى من أجزاء الفصائل classes والدوال methods والمتغيرات variables تكون متاحة

لمستخدم الفصيلة class وأيها غير متاح ، تماماً مثل العمر في النقطة السابقة والذي نريد منع المستخدم من التعامل معه مباشرة.

ولكى تستخدم المعدلات Modifiers فإننا نقوم بوضع المعدل Modifier المطلوب ثم اسم المتغير أو الدالة Method أو الفصيلة Class كما فى الأمثلة التالية:

```
Public class Employee
Private Age As Integer
Public Function Calculate( ) As Double
```

وعلى الرغم من أن استخدام المعدلات Modifiers اختياري ، لكن من المفيد استخدامها لأنه في كثير من الأحيان ، نجد أن ترك المستخدم ليحدد قيم بعض المتغيرات قد ينتج عنه خطأ منطقياً كما في مثال العمر.

وتقوم لغة Visual Basic بتوفير العديد من مستويات المعدلات Modifiers والتي سنقوم بتوضيح بعضها مثل:

1. المعدل العام public.
2. المعدل الخاص private.
3. المعدل المحمي protected.
4. المعدل الصديق Friend.

## أولاً: المعدل العام public:

يتم تعريف العضو بالمعدل العام public إذا أردنا أن نجعل استدعاء هذا العضو ممكناً من أي جزء من أجزاء البرنامج سواء قمنا باستدعاء هذا العضو من داخل الفصيلة Class التي تم تعريفه فيها أم من خارجها.

## ثانياً: المعدل الخاص private:

يتم تعريف العضو بالمعدل الخاص private إذا أردنا أن نجعل استدعاء هذا العضو غير متاح من أي جزء من أجزاء البرنامج إلا من داخل الفصيلة class التي تم تعريف هذا العضو فيها.

## ثالثاً: المعدل المحمي protected:

يتم تعريف العضو بالمعدل المحمي protected إذا أردنا أن نجعل استدعاء هذا العضو متاحاً من داخل الفصيلة class التي تم تعريف هذا العضو فيها أو من داخل أي فصيلة class ترث من الفصيلة class التي تم تعريف هذا العضو فيها.

## رابعاً: المعدل الصديق friend:

يتم تعريف العضو بالمعدل الصديق friend إذا أردنا أن نجعل استدعاء هذا العضو متاحاً من أي جزء من أجزاء البرنامج الحالي.

الجدول التالي يوضح الأربع معدلات modifiers بالطريقة التالية:

إذا قمنا بتعريف عضو وليكن المتغير Age ، ثم قمنا باستخدام أي من المعدلات Modifiers السابق توضيحهم ، فما إمكانية استدعاء هذا المتغير من أي مكان في البرنامج؟

فمثلاً إذا أردنا استدعاء هذا المتغير من نفس الفصيلة Class (انظر الصف الثاني من الجدول) ، فإن استخدام أي من المعدلات Modifiers يجعل عملية الاستدعاء ممكنة.

أما إذا أردنا استدعاء هذا المتغير من أى فصيلة class واقعة فى نفس المجال namespace (انظر الصف الثالث من الجدول) ، فإن استخدام المعدلات Modifiers (friend و public) فقط يجعل عملية الاستدعاء ممكنة ، أما إذا تم

استخدام المعدلات Modifiers (protected و private) فإن عملية الاستدعاء ينتج عنها خطأ في الترجمة Compile Error.

وبالمثل يمكنك متابعة باقي الجدول.

| مستويات معدلات التوصل Modifiers | | | | مكان استدعاء العضو |
|---|---|---|---|---|
| public | protected | private | friend | |
| Yes | Yes | Yes | Yes | من نفس الفصيلة class. |
| Yes | No | No | Yes | من أى فصيلة class واقعة فى نفس المجال namespace. |
| Yes | No | No | No | من أى فصيلة class تقع خارج المجال namespace. |
| Yes | Yes | No | Yes | من فصيلة فرعية Sub class واقعة فى نفس المجال namespace. |
| Yes | Yes | No | No | من فصيلة فرعية sub class واقعة خارج المجال namespace. |

إذن نستطيع الرجوع إلي نقطتنا الأساسية وهي أننا نريد منع المستخدم من التعامل مع المتغير Age مباشرة ، ويمكننا تحقيق هذا المطلب من خلال تعريف هذا المتغير بالمعدل الخاص private Modifier ، ولكن هنا يظهر سؤال أخر: وهو كيف يمكننا تغيير قيمة هذا المتغير إذن؟

في هذه الحالة نقوم بتعريف دالة Method خاصة تسمي بالدالة المغيرة Mutator (وذلك لأنها تقوم بتغيير قيمة المتغير) وهذه الدالة method تستقبل معاملاً parameter من نفس نوع المتغير الذي نريد تغيير قيمته ثم تقوم بتعديل قيمة المتغير بالقيمة الجديدة وهي تأخذ الشكل التالي:

```
Public Function SetAge (a As Integer)
        Age = a
End Function
```

وقد يظهر هنا سؤال هو: ما دامت قيمة المتغير الخاص بالعمر ستتغير سواء باستخدام قيمة المتغير مباشرة أو عن طريق الدالة المغيرة Mutator ، إذن فما الداعي لاستخدام الدالة المغيرة Mutator؟

الإجابة تتمثل في أنه باستخدام الدالة المغيرة Mutator ، نستطيع التأكد من قيمة المتغير عن طريق استخدام الجملة الشرطية IF وبالتالي نضمن عدم وجود قيمة خاطئة لا نريدها. (سنقوم بتوضيح كيفية استخدام الدالة المغيرة Mutator من خلال المثال القادم).

أما إذا أردنا معرفة قيمة هذا المتغير دون تغيير قيمته ، ففي هذه الحالة نقوم بتعريف دالة Method خاصة تسمي بالدالة الموصلة Accessor (وذلك لأنها تقوم بتوصيلنا لقيمة هذا المتغير) وهذه الدالة method لا تستقبل أي معامل parameter وتقوم فقط بإرجاع قيمة المتغير وهي تأخذ الشكل التالي:

```
Public Function GetAge () As Integer
        Return Age
End Function
```

المثال التالي يوضح استخدام المعدلات Modifiers.

## مثال 3: استخدام المعدلات Modifiers:

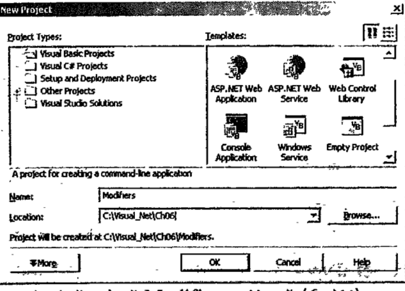
أولاً: هدف المثال:

توضيح كيفية استخدام المعدلات Modifiers.

ثانياً: خطوات إنشاء البرنامج:

1. تأكد أولاً من فتح البرنامج.
2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر لمسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا شاشة إنشاء مشروع جديد New Project.
3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".
4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".
5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Modifiers وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch06" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 6 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




شكل (6 المعدلات Modifiers الخطوة الخامسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السابقة لمزيد من الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Employee كما شرحنا في الفصول السابقة ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:  Public Class Employee
2:     Private salary As Double
3:     Protected name As String
4:     Public university As String
5:
6:     Public Function SetSalary(ByVal s As Double)
7:        If (s > 0) Then
8:           salary = s 'ok, using a private member from the same
    class
9:           Console.WriteLine("Salary is " & salary)
10:
11:       Else
12:          Console.WriteLine("Wrong Value")
13:       End If
14:    End Function
```

```
15:
16:    Public Function GetSalary() As Double
17:       Return salary 'ok, using a private member from the same
   class
18:    End Function
19:
20:    Public Function GetName() As String
21:       Return name 'ok, using a protected member from the same
   class
22:    End Function
23:
24:    Public Function SetName(ByVal n As String)
25:       name = n 'ok, using a protected member from the same class
26:    End Function
27: End Class
28:
29: Public Class Engineer
30:    Inherits Employee
31:    Public Function CalculateTax()
32:       Dim s As Double = GetSalary() 'ok, Engineer is a sub class
33:       'double s1 = salary    'Error since salary is private
34:       university = "Alex" 'ok, using a public member from
   anywhere
35:
36:       Dim Tax As Double = s * 0.1
37:       Console.WriteLine("Tax for" & name & "is " & Tax)
38:       Console.WriteLine("University is " & university)
39:    End Function
40: End Class
41:
42: Class Accountant
43:    Public Function CalculateTax()
44:       'string n = name        'Error since Accountant is not a sub
   class
45:
46:    End Function
47:
48: End Class
```

## ثالثاً: تحليل المثال:

- في السطر رقم 2 يتم استخدام المعدل الخاص private modifier للمتغير salary وهذا يعني أن استخدام هذا المتغير يكون متاحاً فقط من داخل الفصيلة Employee فقط ولذلك فإن السطر رقم 8 لا يسبب أي خطأ لأننا استخدمنا هذا المتغير في الفصيلة Employee وبالمثل نجد أيضاً أن السطر رقم 9 لا يسبب أي خطأ ، أما بالنسبة للسطر رقم 33 فيسبب خطأ في الترجمة Compile وذلك بسبب محاولة استخدام المتغير salary من خارج الفصيلة Employee.
- في السطور من 6 إلي 14 نقوم بإنشاء الدالة المغيرة Mutator لقيمة المتغير salary والتي نقوم فيها بالتأكد من أن قيمة المعامل Parameter لا بد أن تكون موجبة ، وبالمثل يتم تعريف الدالة المغيرة Mutator لقيمة المتغير name في السطور من 24 إلي 26.
- في السطور من 16 إلي 18 نقوم بإنشاء الدالة الموصلة Accessor لقيمة المتغير salary ، وبالمثل يتم تعريف الدالة الموصلة Accessor لقيمة المتغير name في السطور من 20 إلي 22 مع ملاحظة استخدام المعدل العام public modifier مع كلا الدالتين Methods حتي يمكن استخدامهما لتغيير ومعرفة قيمة المتغير من أي فصيلة Class.
- في السطر رقم 3 يتم استخدام المعدل المحمي protected modifier للمتغير name وهذا يعني أن استخدام هذا المتغير يكون متاحاً فقط من داخل الفصيلة Employee أو من داخل أي فصيلة Class ترث من الفصيلة Employee ، ولذلك فإن السطر رقم 21 لا يسبب أي خطأ لأننا استخدمنا هذا المتغير في الفصيلة Employee وبالمثل نجد أيضاً أن السطر رقم 25 لا يسبب أي خطأ كما أن السطر رقم 37 لا يسبب أي خطأ لأن

الفصيلة Engineer ترث من الفصيلة Employee ، أما بالنسبة للسطر رقم 44 فيسبب خطأ في الترجمة Compile وذلك بسبب محاولة استخدام المتغير name من فصيلة Class لا ترث من الفصيلة Employee.

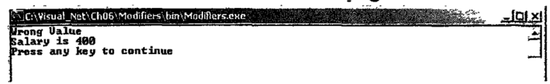
- في السطر رقم 4 يتم استخدام المعدل العام public modifier للمتغير university وهذا يعني أن استخدام هذا المتغير يكون متاحاً من أي مكان ولذلك يمكن استدعاء هذا المتغير من أي مكان بلا مشاكل.

نحتاج الآن لكتابة الأوامر في الدالة الرئيسية Main() ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:     Sub Main()
4:        Dim e As Engineer = New Engineer()
5:        'e.salary = 60           'Error since salary is private
6:        'e.name = "Mostafa"  'Error since name is protected
7:        e.SetName("Mostafa")
8:        e.SetSalary(-300)
9:        e.university = "Alex" 'ok, using a public member from
    anywhere
10:
11:       e.SetName("Kadry")
12:       e.SetSalary(400)
13:       e.university = "Cairo" 'ok, using a public member from
    anywhere
14:    End Sub
15:
16: End Module
```

- في السطر رقم 8 يتم استدعاء الدالة SetSalary() مع تحديد قيمة المرتب ليكون بالسالب ولذلك نجد أن الدالة SetSalary() تطبع القيمة Wrong Value بسبب عدم تحقق شرط أن يكون المرتب موجباً.
- في السطر رقم 12 يتم استدعاء الدالة SetSalary() مع تحديد قيمة المرتب ليكون موجباً ولذلك نجد أن الدالة SetSalary() تطبع قيمة المرتب بسبب تحقق شرط أن يكون المرتب موجباً.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 7.




(شكل 7) المعدلات Modifiers

# الفصل السابع

## تعدد الأشكال والتجريد

## Polymorphism And Abstraction

تعدد الأشكال والتجريد
Polymorphism And Abstraction

في هذا الفصل نستكمل شرح مفاهيم البرمجة موجهــة الهدف (Object-Oriented Programming (OOP حيــث نتعــرف علــي مبــدأ تعــدد الأشــكال Polymorphism والتجريد Abstraction وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. تعدد الأشكال Polymorphism.
3. التحميل الزائد للدوال Method Overloading.
4. نسخ الدوال Method Overriding.
5. التجريد Abstraction.
6. خصائص الفصيلة المجردة Abstract class.

## مقدمة:

- تعرفنا في الفصل السابق على المزيد من مبادئ البرمجة موجهــة الهــدف OOP ونستكمل في هذا الفصل شرح المزيد من هذه المبادئ.
- سوف نتعرف في هذا الفصل علي مبدأ تعــدد الأشــكال Polymorphism والتجريد Abstraction وسنبدأ بــالتعرف علــي مبــدأ تعــدد الأشــكال .Polymorphism

## تعدد الأشكال Polymorphism:

- تعتمد فكرة تعدد الأشكال Polymorphism علي تنفيذ الأوامــر بطريقــة مختلفة علي حسب الموقف ، فمثلاً تجد أن الزر Ok في الهاتف المحمول له أكثر من وظيفة علي حسب الموقف ، بمعني أن هذا الزر يستخدم للرد علي المكالمات عند استقبال المكالمات ، كما يستخدم نفس هــذا الــزر لقــراءة الرسائل القصيرة عند استقبال الرسائل ، كما يستخدم نفس هذا الزر لتغييــر النغمات عند وجود الحاجة لتغيير النغمة ... إلخ.
- أي أن نفس الزر قام بتنفيذ وظيفة مختلفة علي حسب الموقف وهذا هو مبدأ تعدد الأشكال Polymorphism وهو يقوم بعمل تصميم جيــد للبرنــامج ، فبدلاً من إنشاء زر للرد علي المكالمات وزر أخر لقراءة الرسائل القصــيرة وزر أخر لتغيير النغمات ... إلخ ، فيمكن تصميم زر واحد وإعادة استخدامه وبالتالي نحتاج زراً واحداً فقط لتنفيذ الوظائف المختلفة وإلا كنــت ســتجد هاتفك المحمول يحتوي علي ما لا يقل عن 200 زر لتنفيذ جميع وظائفه.
- يوجد شكلان في غاية الأهمية لتنفيذ فكرة تعدد الأشــكال Polymorphism وهما:
    1. التحميل الزائد للدوال Method Overloading.
    2. نسخ الدوال Method Overriding.

## أولاً: التحميل الزائد للدوال Method Overloading:

إن عملية التحميل الزائد للدوال Method Overloading تعني إنشاء فصيلة Class تحتوي علي تعريف أكثر من دالة Method بنفس الاسم في نفس هذه الفصيلة Class ، ولذلك لا بد أن تختلف الدوال Method في واحدة علي الأقل من النقاط التالية (حتي يمكن التمييز بين هذه الدوال Methods وإمكانية استدعاء الدالة Method Calling الصحيحة):

1. عدد المعاملات parameters.
2. نوع معامل parameter data type واحد علي الأقل أو اختلاف ترتيب المعاملات parameters مع اختلاف نوع هذه المعاملات parameters.

ملحوظة:

إذا اختلفت دالتان Methods في نوع القيمة المرتجعة Return Value فقط ، فلا يمكن استخدامهما لتحقيق مبدأ التحميل الزائد للدوال Method Overloading ، فمثلاً نفترض أننا أنشأنا دالة Method تسمي Calculate() بالشكل التالي:

```
Public Function Calculate(x As Double) As Double
```

ثم أنشأنا دالة Method أخري بالشكل التالي:

```
Public Function Calculate(x As Double) As Float
```

فلا يمكننا استخدام هاتين الدالتين في نفس الفصيلة class وإلا نتج عن ذلك حدوث خطأ في الترجمة Compile.

كما يوجد أيضاً خطأ شائع في عملية التحميل الزائد للدوال Method Overloading عند محاولة استخدام درال Method تختلف فقط في اسم معامل Parameter هذه الدوال Methods ، فمثلاً الدالتان Method:

```
Public Function Calculate(x As Double) As Double
Public Function Calculate(y As Double) As Double
```

لا يصح استخدامهما في فصيلة class واحدة لتحقيق التحميــل الزائــد للــدوال Method Overloading ، فهاتان الدالتان Methods متماثلتان تماماً لأن اسم المعامل Parameter لا يؤدي إلي اختلاف الدالة Method ففي النهاية نجد أن هاتين الدالتين Methods تستقبلان قيمة من نوع Double وبالتــالي لا يمكــن التمييز بينهما علي الرغم من اختلاف اسم المعامل Parameter.

الدوال التاليــة تعتبــر مثــالاً صــحيحاً للتحميــل الزائــد للــدوال Method Overloading:

```
Public Function Calculate(x As Float) As Double
Public Function Calculate(y As Float , z As Double) As Float
Public Function Calculate(z As Double,y As Float) As Double
```

حيث تختلف الدالة Method الأولي عن الثانية والثالثة فــي عــدد المعــاملات Parameters وتختلف الدالة Method الثانية عن الثالثة في ترتيب المعــاملات Parameters ونوعها.

المثال التالي يوضح استخدام التحميل الزائد للدوال Method Overloading.

## مثال 1: التحميل الزائد للدوال Method Overloading:

أولاً: هدف المثال:

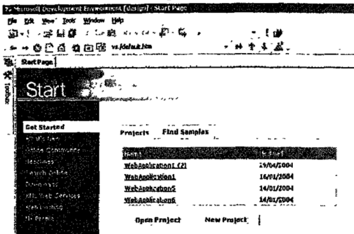
هذا المثال يوضح لنا مفهوم التحميل الزائــد للــدوال Method Overloading. حيث سنقوم بإنشاء دالتين Methods بحيث تمثل كلا الدالتين Methods وظيفة الأزرار في الجهاز المحمول حيث يمكن عن طريق نفس الزر كتابــة أرقــام أو حروف ، ونريد أن تقوم كل من الدالتين Methods بطباعة الــرقم أو الحــرف الذي تم إدخاله.

ثانياً: خطوات إنشاء البرنامج:

1. افتح البرنامج عن طريق اختيار

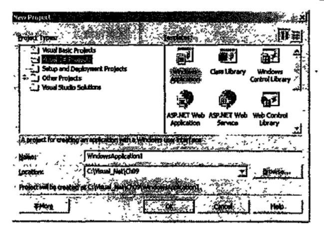
Start → Programs → Microsoft Visual Studio.Net → Microsoft Visual Studio.Net

لتظهر لك نافذة البرنامج كما هو واضح في شكل 1.



شكل 1) التحميل الزائد للدوال Method Overloading الخطوة الأولى)

2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا الشاشة كما هو واضح في شكل 2.



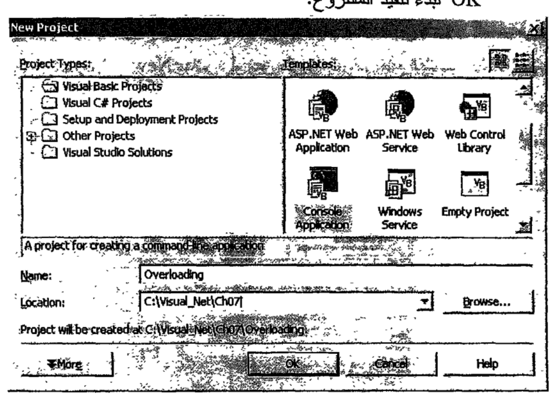
شكل 2) التحميل الزائد للدوال Method Overloading الخطوة الثانية)

3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".

4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".

5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Overloading وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch07" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 3 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




شكل 3) التحميل الزائد للدوال Method Overloading الخطوة الخامسة)

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السابقة لمزيد من الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Mobile كما شرحنا في الفصول السابقة ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:  Public Class Mobile
2:      Public Function Click(ByVal n As Integer)
3:          Console.WriteLine("I am a number " & n)
4:      End Function
5:
6:      Public Function Click(ByVal c As Char)
7:          Console.WriteLine("I am a character " & c)
8:      End Function
9:
10: End Class
```

## ثالثاً: تحليل المثال:

- في السطور من 2 إلي 4 يتم تعريف الدالة Click( ) والتي تستقبل معاملاً parameter من نوع Integer وهذه الدالة Method تقوم بطباعة الرقم الذي أدخله المستخدم.
- في السطور من 6 إلي 8 يتم استخدام نفس اسم الدالة Click( ) لتعريف دالة Method أخري تستقبل معاملاً parameter من نوع Char وهذه الدالة Method تقوم بطباعة الحرف الذي أدخله المستخدم.
- وفي استخدام هاتين الدالتين Method تحقيق لمبدأ التحميل الزائد Overloading حيث اختلفت هاتين الدالتين Methods في نوع المعامل Parameter.

نحتاج الآن لكتابة الأوامر في الدالة الرئيسية ()Main ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:     Sub Main()
4:        Dim m As Mobile = New Mobile()
5:        Dim x As Integer = 1
6:        Dim y As Char = "a"
7:        m.Click(x)
8:        m.Click(y)
9:
10:    End Sub
11:
12: End Module
```

- في السطر رقم 4 يتم إنشاء هدف object من نوع الفصيلة Mobile.
- في السطر رقم 5 يتم تعريف متغير اسمه x من نوع Integer.
- في السطر رقم 6 يتم تعريف متغير اسمه y من نوع Char.
- في السطر رقم 7 يتم استدعاء الدالة ( )Click وتمرير المتغير x كمعامل Parameter لها والذي من نوع Integer وينتج عن ذلك استدعاء الدالة Method Calling المعرفة في السطر رقم 2 من الملف السابق لأنها الوحيدة التي تستقبل معاملاً Parameter من نوع Integer.
- في السطر رقم 8 يتم استدعاء الدالة ( )Click وتمرير المتغير y كمعامل Parameter لها والذي من نوع Char وينتج عن ذلك استدعاء الدالة Method Calling المعرفة في السطر رقم 6 من الملف السابق لأنها الوحيدة التي تستقبل معاملاً Parameter من نوع Char.

- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 4.

```
C:\Visual_Net\Ch07\Overloading\bin\Overloading.exe
I am a number 1
I am a character a
Press any key to continue_
```

شكل 4) تنفيذ البرنامج)

يمكنك إغلاق البرنامج الآن عن طريق فتح قائمة File ثم Close Solution استعداداً للمثال التالي.

## ثانياً: نسخ الدوال Method Overriding:

تعرفنا في الفصل السابق علي مبدأ الوراثة Inheritance وتعلمنا كيفية تنفيذ الوراثة Inheritance من خلال كود البرمجة ، ويعتمد مبدأ نسخ الدوال Method Overriding علي مبدأ الوراثة Inheritance في التنفيذ ولذلك تعال معي نتخيل الموقف الآتي.

افترض وجود شخص له سلطة اتخاذ القرار ، كما يوجد مدير له أيضاً سلطة اتخاذ القرار ، وبفرض أن المدير يرث صفاته من الشخص ، فإذا أردنا تنفيذ قرار معين ، فهل سيتم تنفيذ قرار الشخص أم المدير؟

ولتوضيح تلك المشكلة من خلال كود البرمجة ، فتعال نفترض وجود فصيلة Class تسمي Person وهذه الفصيلة Class تمثل الفصيلة الأم Parent class بحيث تحتوي علي الدالة MakeDecision( ).

سنفترض أيضاً وجود فصيلة Class أخري باسم Manager ترث من الفصيلة Person ثم قمنا بإنشاء الدالة MakeDecision( ) في الفصيلة Manager مع ملاحظة أن الفصيلة Manager تحتوي علي الدالة MakeDecision( ) من الفصيلة Person تحقيقاً لمبدأ الوراثة Inheritance.

المشكلة تظهر كالآتي: افترض أننا أنشأنا هدفاً object من نوع الفصيلة Manager ثم قمنا باستدعاء الدالة MakeDecision( ) ، فهنا يظهر سؤال: أي دالة Method سيتم استدعاءها Calling ، هل هي الدالة Method الموجودة في الفصيلة Person أم الفصيلة Manager ، أو بمعني أخر إذا أردنا تنفيذ قرار معين ، فهل سيتم تنفيذ قرار الشخص أم قرار المدير ، وبطبيعة الحال فإن قرار كل منهما يختلف عن الأخر ، وللإجابة عن هذا السؤال فلا بد من فهم مبدأ نسخ الدوال Method Overriding.

يمكننا تعريف مبدأ نسخ الدوال Method Overriding كالتالي:

نسخ الدوال Method Overriding يعني وجود دالة Method معينة تم تعريفها في الفصيلة الأم Parent class ثم يتم استخدام نفس اسم الدالة Method بنفس نوع المتغيرات وترتيبها وعددها في الفصيلة الفرعية Sub class.

إذن ستكون هناك دالة Method تسمي MakeDecision( ) في الفصائل Person و Manager وهنا يظهر سؤال: كيف يتم معرفة الدالة Method الصحيحة التي نريد استدعاءها؟

بمعني: إذا أنشأنا هدفاً object من نوع الفصيلة Person ثم استدعينا Call الدالة MakeDecision( ) ، فهل سيتم استدعاء الدالة MakeDecision( ) الخاصة بالفصيلة Person أم الخاصة بالفصيلة Manager؟

الإجابة: يتم استدعاء الدالة الصحيحة علي حسب نوع الهدف object الذي أنشأناه ، فمثلاً إذا كتبنا السطور التالية:

```
Dim p As Person = New Person()
p.MakeDecision()
```

ففي هذه الحالة يتم استدعاء الدالة MakeDecision( ) الخاصة بالفصيلة Person لأن الهدف object الذي أنشأناه من نوع الفصيلة Person.

اما في السطور التالية:

```
Dim m As Manager = New Manager()
m.MakeDecision()
```

ففي هذه الحالة يتم استدعاء الدالة ( )MakeDecision الخاصة بالفصيلة Manager لأن الهدف object الذي أنشأناه من نوع الفصيلة Manager.

أما في السطور التالية:

```
Dim pm As Person = New Manager()
pm.MakeDecision()
```

ففي هذه الحالة يتم استدعاء الدالة ( )MakeDecision الخاصة بالفصيلة Manager لأن الهدف object الذي أنشأناه من نوع الفصيلة Manager.

ولفهم السطرين السابقين من الناحية المنطقية ، فيمكننا القول أن هذا الشخص له صلاحيات المدير ويتصرف مثله ولذلك تم تنفيذ قرار المدير وبالتالي تمت عملية نسخ overriding لقرار الشخص بقرار المدير.

أما من ناحية البرمجة ، فإن الهدف object الذي أنشأناه هو من نوع الفصيلة Manager ولذلك يتم استدعاء الدالة ( )MakeDecision الخاصة بالفصيلة Manager تحقيقاً لمبدأ نسخ الدوال Method Overriding السابق شرحه.

ولكي تتم عملية نسخ الدوال Method Overriding ، فلا بد من تعريف الدالة MakeDecision() الموجودة في الفصيلة Person بالكلمة Keyword المعروفة باسم "يمكن نسخها" Overridable.

إذن الدوال الممكن نسخها Overridable Methods هي الدوال Methods التي يتم إجراء عملية نسخ Override لها في الفصائل الفرعية Sub classes ويتم استدعاء الدالة Method Calling الصحيحة علي حسب نوع الهدف object الذي ننشئه.

ويعتبر شرط عملية النسخ Overriding في الفصائل الفرعية Sub classes ، أن تتم إعادة كتابة الدالة الجديدة بشرط استخدام نفس اسم الدالة Method بنفس نوع المتغيرات وترتيبها وعددها مع استخدام الكلمة keyword المعروفة باسم Overrides قبل اسم الدالة Method.

المثال التالي يوضح عملية نسخ الدوال Method Overriding.

## مثال 2: نسخ الدوال Method Overriding:

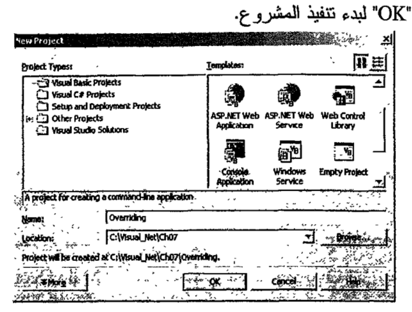
أولاً: هدف المثال:

توضيح كيفية تنفيذ مبدأ نسخ الدوال Method Overriding.

ثانياً: خطوات إنشاء البرنامج:

1. تأكد أولاً من فتح البرنامج.
2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا شاشة إنشاء مشروع جديد New Project.
3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون "Visual Basic Projects".
4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".
5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Overriding وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch07" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 5 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الـزر "OK" لبدء تنفيذ المشروع.




(شكل 5) نسخ الدوال Method Overriding الخطوة الخامسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبـرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السـابقة لمزيـد مـن الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Person كما شرحنا فـي الفصـول السابقة ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:  Public Class Person
2:     Public Overridable Function MakeDecision()
3:        Console.WriteLine("Person Decision")
4:     End Function
5:
6:  End Class
7:
8:  Class Manager
9:     Inherits Person
10:    Public Overrides Function MakeDecision()
11:       Console.WriteLine("Manager Decision")
12:    End Function
```

```
13: End Class
```

## ثالثاً: تحليل المثال:

- في السطر رقم 2 يتم تعريف الدالة MakeDecision() ونريد أن نجعل هذه الدالة Method قابلة لإجراء عملية النسخ Overriding ولذلك تم تعريف الدالة MakeDecision() كدالة يمكن نسخها Overridable.
- في السطر رقم 8 يتم تعريف الفصيلة Manager والتي ترث من الفصيلة Person.
- في السطر رقم 10 يتم إجراء عملية نسخ Overriding للدالة MakeDecision() وتلاحظ استخدام كلمة Overrides لتدل علي عملية النسخ Overriding وتلاحظ أيضاً استخدام تعريف الدالة MakeDecision() كما كان بالضبط في الفصيلة Person (راجع السطر رقم 2).

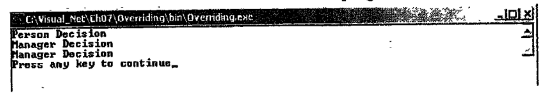
نحتاج الآن لكتابة الأوامر في الدالة الرئيسية Main() ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:     Sub Main()
4:        Dim p As Person = New Person()
5:        p.MakeDecision()
6:
7:        Dim m As Manager = New Manager()
8:        m.MakeDecision()
9:
```

```
10:         Dim pm As Person = New Manager()
11:         pm.MakeDecision()
12:
13:     End Sub
14:
15: End Module
```

- في السطر رقم 4 يتم إنشاء هدف object من نوع الفصيلة Person.
- في السطر رقم 5 يتم استدعاء الدالة ()MakeDecision وينتج عن ذلك استدعاء الدالة ()MakeDecision المعرفة في الفصيلة Person لأن الهدف object المسمي p من نوع الفصيلة Person.
- في السطر رقم 7 يتم إنشاء هدف object من نوع الفصيلة Manager.
- في السطر رقم 8 يتم استدعاء الدالة ()MakeDecision وينتج عن ذلك استدعاء الدالة ()MakeDecision المعرفة في الفصيلة Manager لأن الهدف object المسمي m من نوع الفصيلة Manager.
- في السطر رقم 10 يتم إنشاء هدف object من نوع الفصيلة Manager.
- في السطر رقم 11 يتم استدعاء الدالة ()MakeDecision وينتج عن ذلك استدعاء الدالة ()MakeDecision المعرفة في الفصيلة Manager لأن الهدف object المسمي pm من نوع الفصيلة Manager.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 6.




شكل 6) نسخ الدوال Method Overriding)

## التجريد Abstraction:

مبدأ التجريد Abstraction هو المبدأ الذي يصف الأشياء بتركيز علي المدخلات inputs والمخرجات outputs بدلاً من التركيز علي تفاصيل التنفيذ.

بمعني أخر ، فإن مبدأ التجريد Abstraction هو كيفية وصف الأشياء بأبسط صورة ممكنة ، فمثلاً نستطيع وصف الانسان بأنه مخلوق عاقل يفكر وهذا يعني أننا نركز علي التفاصيل الأساسية للانسان دون تحديد طريقة التفكير التي تختلف من شخص لأخر ، أي أننا ركزنا علي صفة التفكير دون التركيز علي تفاصيل كيفية تنفيذ التفكير.

ويتم تنفيذ مبدأ التجريد Abstraction في البرمجة عن طريق تجميع المتغيرات والدوال Methods الأساسية الموجودة في فصائل classes معينة ، ويتم تجميعهم في فصيلة class منفصلة وهذه الفصيلة class تعتبر في هذه الحالة غير كاملة لأنها تحتوي علي التفاصيل الأساسية فقط ولا تحتوي علي جميع التفاصيل وتعرف في هذه الحالة بالفصيلة المجردة Abstract class.

## خصائص الفصيلة المجردة Abstract class:

1. يكون شكل الإعلان عنها كالتالى:

```
Public MustInherit class Person
        //Code of the class
End Class
```

2. تحتوي علي التفاصيل الأساسية فقط والمشتركة بين العديد من الفصائل Classes.

3. لا يمكنك إنشاء هدف object من الفصيلة المجردة Abstract class وذلك لأنها غير كاملة التفاصيل.

4. يتم تكملة التفاصيل الناقصة للفصيلة المجردة Abstract class عن طريق إنشاء فصيلة class جديدة ترث من الفصيلة المجردة Abstract class وتحتوي علي باقي التفاصيل.

المثال التالي يوضح استخدام الفصيلة المجردة Abstract class.

## مثال 3: التجريد Abstraction:

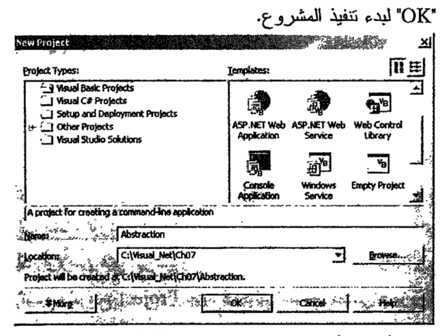
أولاً: هدف المثال:

توضيح كيفية تنفيذ مبدأ التجريد Abstraction.

ثانياً: خطوات إنشاء البرنامج:

1. تأكد أولاً من فتح البرنامج.
2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا شاشة إنشاء مشروع جديد New Project.
3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".
4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".
5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Abstraction وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch07" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 7 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الـزر "OK" لبدء تنفيذ المشروع.




شكل 7) التجريد Abstraction الخطوة الخامسة

تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبـرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السـابقة لمزيـد مـن الشرح.

قم الآن بإضافة فصيلة Class جديدة اسمها Person كما شرحنا فـي الفصـول السابقة ثم قم بتعديل كود البرمجة ليصبح كالتالي:

```
1:    Public MustInherit Class Person
2:        Public Name As String
3:        Public Age As Integer
4:
5:        Public Function Run()
6:            Console.WriteLine("I Run")
7:        End Function
8:
9:        Public Function Swim()
10:           Console.WriteLine("I Swim")
11:       End Function
```

```
12:
13:     Public Function PlayFootball()
14:        Console.WriteLine("I Play")
15:     End Function
16:
17:  End Class
18:
19:  Class Engineer
20:     Inherits Person
21:     Public Function Think()
22:        Console.WriteLine("An Engineers Thinks")
23:     End Function
24:
25:  End Class
```

## ثالثاً: تحليل المثال:

- في السطور من 1 إلي 17 يتم تعريف الفصيلة Person ولا جديد هنا مع ملاحظة أن الفصيلة Person هي فصيلة مجردة Abstract class.
- في السطر رقم 19 يتم تعريف الفصيلة Engineer وهذه الفصيلة Class ترث من الفصيلة Person وهذا مسموح به لأن الفصائل المجردة Abstract classes لا تمنع الوراثة Inheritance منها.
- في السطور من 21 حتي 23 يتم تعريف الدالة Think().

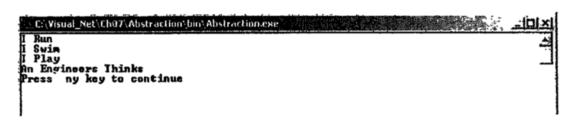
نحتاج الآن لكتابة الأوامر في الدالة الرئيسية Main() ويتم ذلك عن طريق الرجوع إلي التبويب Tab المسمي Module1.vb من أعلي نافذة المشروع Project.

يمكنك تعديل كود البرمجة ليصبح كالتالي.

```
1:  Module Module1
2:
3:     Sub Main()
```

```
4:      'Dim p As Person = New Person()    'error since Person is
   abstract
5:
6:      Dim e As Engineer = New Engineer()
7:      e.Run()
8:      e.Swim()
9:      e.PlayFootball()
10:     e.Think()
11:
12:   End Sub
13:
14: End Module
```

- في السطر رقم 4 يتم توضيح خطأ واضحاً حيث تم إنشاء هدف Object من الفصيلة Person وهذا غير مسموح به لأن الفصائل المجردة Abstract classes لا يمكنك إنشاء هدف object منها ولذلك تم وضع هذا السطر كتعليق Comment.
- في السطر رقم 6 يتم إنشاء هدف object من نوع الفصيلة Engineer.
- في السطور من 7 إلي 10 يتم استدعاء دوال الفصيلة Engineer ولاجديد هنا.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 8.




شكل 8) التجريد Abstraction)

# الفصل الثامن

## الاستثناءات

## Exceptions

في هذا الفصل نتعرف علـــي موضـــوع الاســـتثناءات Exceptions كما نتعرف علي كيفية معالجة الاستثناء Exception Handling وذلك مـــن خـــلال النقـــاط التالية:

1. مقدمة Introduction.
2. الاستثناء Exception.
3. معالجة الاستثناءات Exceptions Handling.
4. جملة try و catch.
5. جملة finally.

الاستثناءات
Exceptions

## مقدمة:

- يوجد دائماً طرفان لأي تطبيق يتم تنفيذه حيث أن الطرف الأول هو المبرمج والطرف الثاني هو مستخدم البرنامج.
- يحاول المبرمج دائماً وضع جميع احتمالات خطأ المستخدم حتي يستمر تنفيذ البرنامج بشكل طبيعي لباقي وظائفه وإلا توقف البرنامج عن العمل.
- ولكن مهما حاول المبرمج تجنب أخطاء المستخدم ، فدائماً توجد بعض الأخطاء الاستثنائية التي تكون بعيدة عن تفكير المبرمج وذلك لأن احتمالات الخطأ لا نهائية ، ولذلك فإننا نحتاج إلي معرفة الاستثناء Exception.

## الاستثناء Exception:

- الاستثناء Exception هو مؤشر لحدوث مشكلة عند تنفيذ أمر معين في البرنامج. وجاءت تسمية الاستثناء Exception بسبب أن الأمر الذي سبب المشكلة يتم تنفيذه بشكل طبيعي ، ولكنه يسبب مشكلة مع بيانات معينة ، أي أن الجملة يتم تنفيذها بشكل طبيعي ويعتبر حدوث مشكلة في هذه الجملة هو حدث استثنائي.
- يعتبر أبسط مثال للاستثناء Exception هو محاولة قسمة رقمين علي بعضهما البعض ، فنجد أن هذه الجملة يتم تنفيذها بشكل طبيعي ولا يمكنك تحديد خطأ معين في عملية القسمة ، ولكن ماذا لو حاول المستخدم قسمة الرقم 10 علي صفر؟
- فكما نعرف من علم الرياضة ، فإن قسمة أي رقم علي الصفر غير مسموح بها ، وبالتالي يعتبر حدوث مشكلة في هذه الجملة هو حدث استثنائي فليس من المعقول أن المستخدم يقوم بالقسمة علي الصفر أغلب الوقت ، ولكنه يقوم بتنفيذ عملية القسمة بشكل طبيعي مع احتمال وقوعه في خطأ القسمة علي الصفر ، أي أن القسمة علي الصفر هو بالفعل حدث استثنائي.

- والأمثلة علي الأحداث الاستثنائية كثيرة ، فمثلاً قد يحاول المستخدم فتح ملف لقراءة بعض البيانات منه ، فنجد أنه من الممكن أن يكون الملف تالفاً بسبب وجود فيروس Virus مثلاً أو قد يكون القرص الصلب Hard Disk بـه بعض العيوب مثل وجود قطاع تالف Bad Sector عليه.
- وللتغلب علي هذه المشاكل ، فإننا نحتاج إلي معالجة الاستثناءات Exceptions Handling.

## معالجة الاستثناءات Exceptions Handling:

- إن عملية معالجة الاستثناءات Exceptions Handling تهدف إلي محاولة اكتشاف الأخطاء الاستثنائية في البرنامج مع معالجة هذه الأخطاء حتي لا يؤدي الاستثناء Exception إلي توقف البرنامج عن العمل ، كما يمكن معرفة بعض المعلومات عن الاستثناء Exception حتي يمكن مستقبلاً تفادي وقوع مثل هذا الاستثناء Exception.
- ويتم عملية معالجة الاستثناءات Exceptions Handling بمنتهي البساطة بأن يتم كتابة جملة Try (وهي أحد الكلمات المحجوزة Reserved Words في لغة Visual Basic والتي تقوم بمعالجة الاستثناءات Exceptions Handling) ثم يتم وضع جميع الأوامر التي يوجد احتمال لوقوع خطأ فيها في جملة Try ثم يليها جملة Catch والتي تقوم بإنشاء رد فعل البرنامج للخطأ الذي نشأ ، فإن حدث خطأ ، فإنه يتم تنفيذ الأوامر في جملة Catch ، وإن لم يحدث خطأ فإن تنفيذ البرنامج يستمر بصورة طبيعية.
- معالجة الاستثناءات Exceptions Handling تأخذ الشكل العام التالي:

```
Try
        'write here all statements
Catch
        'code to handle the exception
```

```
End Try
'Rest of the application
```

- حيث يتم تنفيذ مجموعة الأوامر في جملة Try ، فإذا حدث استثناء Exception من مجموعة الجمل الموجودة داخل جملة Try ، ففي هذه الحالة يتم تنفيذ مجموعة الأوامر الموجودة في جملة Catch لمعالجة الاستثناء Exception الناتج.
- وتتوافر في لغة Visual Basic العديد من الاستثناءات Exceptions المبنية في اللغة حتي توفر علي المبرمج وقتاً كبيراً لمحاولة كتابة كود برمجة لكل خطأ محتمل.
- ومن أشهر الاستثناءات Exceptions هو محاولة القسمة علي صفر والذي سنقوم بتوضيحه في المثال التالي للتعرف علي كيفية التعامل مع الاستثناءات Exceptions.

## مثال 1: الاستثناءات Exceptions:

**أولاً: هدف المثال:**

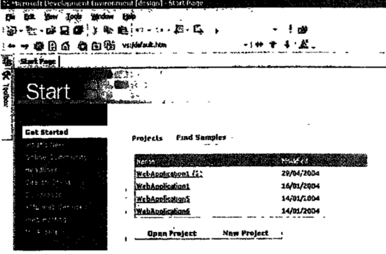
هذا المثال يوضح لنا أهمية التعامل مع الاستثناءات Exceptions عن طريق إنشاء ملف يحتوي علي بعض الأخطاء المحتملة بدون أي معالجة حتي يتبين لنا أهمية موضوع معالجة الاستثناءات Exceptions Handling.

**ثانياً: خطوات إنشاء البرنامج:**

1. افتح البرنامج عن طريق اختيار

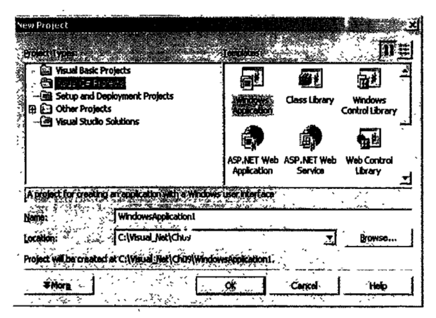
Start → Programs → Microsoft Visual Studio.Net → Microsoft Visual Studio.Net

لتظهر لك نافذة البرنامج كما هو واضح في شكل 1.




شكل (1 الاستثناءات Exceptions الخطوة الأولي

2. لإنشاء مشروع Project جديد فيمكنك الضغط علـى الـزر المسـمى New Project مباشرة أو عن طريق القوائم بفتح القائمـة File ثـم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضـغط على الأزرار Ctrl + Shift + N لتظهر لنا الشاشة كما هو واضح في شكل 2.



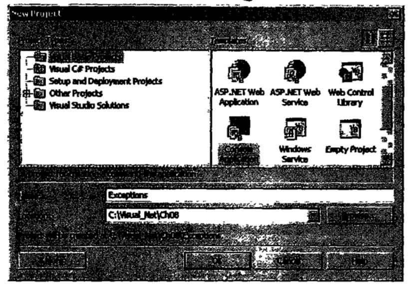

شكل (2 الاستثناءات Exceptions الخطوة الثانية

3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".

4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Console Application".

5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Exceptions وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch08" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 3 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.




(شكل 3) الاستثناءات Exceptions الخطوة الخامسة

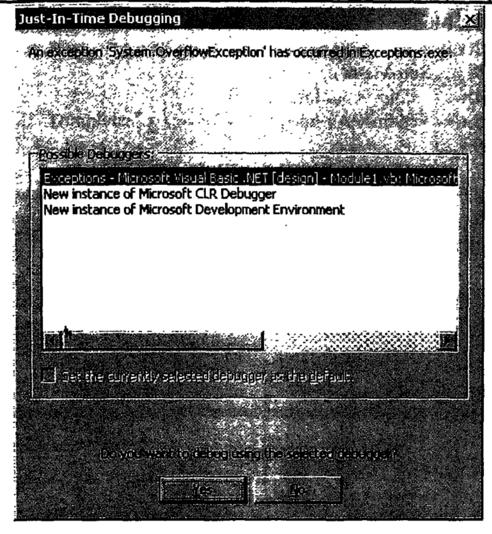
تقوم لغة Visual Basic بإنشاء كود برمجة تلقائياً بدون أي تدخل من المبرمج لتسهيل كتابة كود البرمجة ويمكنك الرجوع إلي الفصول السابقة لمزيد من الشرح.

يمكنك تعديل كود البرمجة ليصبح كالتالي:

```
1:  Module Module1
2:
3:     Sub Main()
4:        Dim x As Integer = 5
5:        Dim y As Integer = 0
6:
7:        Console.WriteLine("Before Exception")
8:
9:        Dim z As Integer = x / y
10:       Console.WriteLine("z = " & z)
11:
12:       Console.WriteLine("After Exception")
13:
14:    End Sub
15:
16: End Module
```

**ثالثاً: تحليل المثال:**

- في السطر رقم 4 يتم تعريف المتغير x.
- في السطر رقم 5 يتم تعريف المتغير y.
- في السطر رقم 7 يتم طباعة رسالة علي شاشة الدوس Dos.
- في السطر رقم 9 يتم قسمة المتغير x علي y ووضع الناتج في المتغير z.
- في السطر رقم 10 يتم طباعة قيمة المتغير z.
- في السطر رقم 12 يتم طباعة رسالة علي شاشة الدوس Dos.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 4.




(شكل 4) تنفيذ البرنامج

- في هذه الشاشة يظهر لنا اسم الاستثناء Exception الذي حدث وهو OverflowException كما توقعنا ، وعند الضغط علي الزر Yes تظهر لنا رسالة نضغط فيها علي الزر Ok لتظهر لنا رسالة أخري تبين اسم الاستثناء Exception الذي حدث ويمكنك الضغط علي الزر Break حتي يبين لنا البرنامج الموضع الذي حدث فيه الاستثناء Exception (وهو بالطبع السطر رقم 9) أو يمكنك الضغط علي الزر Continue للرجوع إلي شاشة البرنامج حتي يمكننا التعديل في البرنامج.

ملحوظة:

- يمكنك فتح قائمة Debug ثم اختيار Stop Debugging لإيقاف إظهار الرسائل التي تبين حدوث الاستثناء Exception وذلك في حالة ضغطك علي الزر Break في الخطوة السابقة.
- بعد انتهاء ظهور الرسائل التي تبين حدوث الاستثناء Exception ، ستجد أن شاشة الدوس Dos تقوم بإظهار الرسالة كما هو واضح في شكل 5.

```
C:\Visual_Net\Ch08\Exceptions\bin\Exceptions.exe
Before Exception

Unhandled Exception: System.OverflowException: Arithmetic operation resulted in
an overflow.
   at Exceptions.Module1.Main() in C:\Visual_Net\Ch08\Exceptions\Module1.vb:line
9
Press any key to continue
```

شكل 5) تنفيذ البرنامج)

- نلاحظ هنا طباعة الرسالة "Before Exception" ثم توقف البرنامج عن تنفيذ باقي الأوامر حيث لم يطبع قيمة المتغير z أو الرسالة "After Exception" وبالتالي تحدث خسائر بالنسبة للشركة التي تستخدم البرنامج بسبب وجود بعض البرامج التي لا تحتمل التوقف عن العمل مثل أي برنامج في مستشفي أو شركة طيران أو هيئة سكة حديدية أو أي برنامج عسكري ، وبالتالي تصبح مهمتنا الآن هي كيفية معالجة الاستثناء Exception الناشئ حتي لا يتم إيقاف تنفيذ البرنامج كما يتضح لنا من المثال التالي.

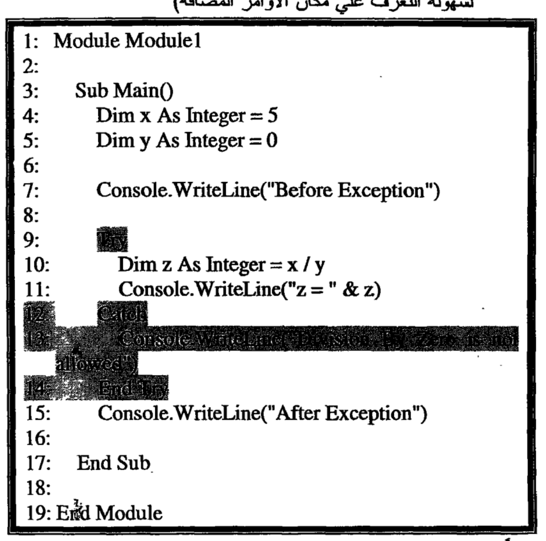
## مثال 2: معالجة الاستثناء Exception Handling:

أولاً: هدف المثال:

توضيح كيفية معالجة الاستثناء Exception Handling لمنع إيقاف تنفيذ البرنامج في المثال السابق.

ثانياً: خطوات إنشاء البرنامج:

1. تأكد أولاً من فتح الملف السابق إنشاؤه في المثال السابق.
2. يمكنك تعديل كود البرمجة ليصبح كالتالي: (تم تظليل الكود الجديد لسهولة التعرف علي مكان الأوامر المضافة)

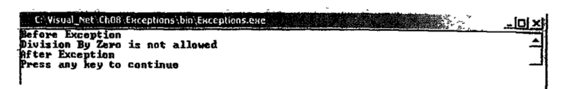


```
1:  Module Module1
2:
3:      Sub Main()
4:          Dim x As Integer = 5
5:          Dim y As Integer = 0
6:
7:          Console.WriteLine("Before Exception")
8:
9:          Try
10:             Dim z As Integer = x / y
11:             Console.WriteLine("z = " & z)
12:         Catch
13:             Console.WriteLine("Division By Zero is not
    allowed")
14:         End Try
15:         Console.WriteLine("After Exception")
16:
17:     End Sub
18:
19: End Module
```

## ثالثاً: تحليل المثال:

- تم وضع جملة القسمة داخل جملة Try ثم يتبعها جملة Catch بحيث أن الاستثناء Exception الممكن حدوثه هو محاولة القسمة علي صفر كما سبق لنا توضيحه.

■ يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علـــي الشاشة كما هو واضح في شكل 6.




(شكل 6) معالجة الاستثناء Exception Handling

■ يتضح لنا من شكل 6 عدم ظهـــور أي رســـائل لتبـــين حـــدوث اســـتثناء Exception كما تمت طباعـــة الرســـالة "Division By Zero is not allowed" والتي كتبناها داخل جملة catch كما تمـــت طباعـــة الرســـالة "After Exception" والتي تبين لنا أن البرنامج لم يتوقف عـــن العمـــل واستكمل تنفيذ أوامره بشكل طبيعي -علي الرغم مـــن حـــدوث الاســـتثناء Exception- وذلك بعكس المثال السابق.

## جملة Finally:

في كثير من الأحيان نحتاج إلي تنفيذ مجموعة معينة من الأوامر – سواء حـــدث استثناء Exception أم لا – ؛ فمثلاً إذا قمنا بإنشاء برنامج يقوم بفـــتح ملـــف ، فإننا لا بد أن نقوم بإغلاق هذا الملف حتي يتم إخلاء الذاكرة من بيانات الملف ، وفي حالة حدوث استثناء Exception ، فلا بد أيضاً من إغلاق الملف لإخـــلاء الذاكرة ، وهذا يعني أنه سواء حدث استثناء Exception أم لم يحدث فإننا نريد تنفيذ مجموعة الأوامر التي تقوم بإغلاق الملف وهذه هي مهمة جملة Finally.

وجملة Finally تأخذ الشكل العام التالي:

```
Finally
        //code to execute
```

حيث يتم كتابة جملة Finally بعد جملتي Try و Catch وفيها يتم كتابة جميع الأوامر التي نريد تنفيذها سواء حدث استثناء Exception أم لا كما يتضح لنا من المثال التالي.

## مثال 3: جملة Finally:

أولاً: هدف المثال:

توضيح كيفية استخدام جملة Finally لمعالجة الاستثناء Exception Handling في المثال السابق.

ثانياً: خطوات إنشاء البرنامج:

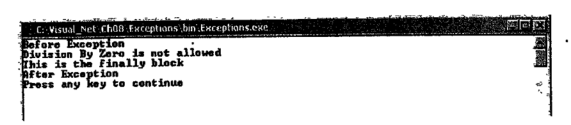
1. تأكد أولاً من فتح الملف السابق إنشاؤه في المثال السابق.
2. يمكنك تعديل كود البرمجة ليصبح كالتالي: (تم تظليل الكود الجديد لسهولة التعرف علي مكان الأوامر المضافة)

```
1:  Module Module1
2:
3:    Sub Main()
4:       Dim x As Integer = 5
5:       Dim y As Integer = 0
6:
7:       Console.WriteLine("Before Exception")
8:
9:       Try
10:         Dim z As Integer = x / y
11:         Console.WriteLine("z = " & z)
12:      Catch
```

```
13:          Console.WriteLine("Division By Zero is not
    allowed")
14:        Finally
15:           Console.WriteLine("This is the finally block")
16:        End Try
17:        Console.WriteLine("After Exception")
18:
19:    End Sub
20:
21: End Module
```

## ثالثاً: تحليل المثال:

- تمت إضافة جملة Finally بعد جملتي Try و Catch لطباعة رسالة محددة حيث سيتم تنفيذ أوامر جملة Finally سواء حدث استثناء Exception أم لا.
- يمكنك تنفيذ البرنامج الآن بالضغط علي الزرين Ctrl + F5 لتحصل علي الشاشة كما هو واضح في شكل 7.




(شكل 7) جملة Finally

يتضح لنا من شكل 7 طباعة الرسالة "This is the finally block" والتي كتبناها في جملة Finally.

تمرين للقارئ:

قم بتغيير قيمة المتغير y في السطر رقم 5 إلي أي قيمة غير صفرية ولتكن 5 ثم قم بتنفيذ البرنامج للتأكد من طباعـــة الرســـالة "This is the finally block" حتي في حالة عدم حدوث استثناء Exception.

# الفصل التاسع

## بناء واجهة المستخدم الرسومية

## Building Graphical User Interface (GUI)




فى هذا الفصل سوف نتناول مكتبة الأدوات التي توفرها لنا لغة Visual Basic والتي تمكننا من بناء واجهة المستخدم الرسومية (GUI) حيث توفر لنا لغة Visual Basic العديد من الأدوات Controls التي سوف نتعرض لشرح بعضها في هذا الفصل وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. إنشاء الواجهة الرسومية GUI.
3. أداة النموذج Form.
4. أداة العنوان Label.
5. أداة النص TextBox.
6. أداة قائمة الاختيارات ComboBox.
7. أداة زر الراديو RadioButton.
8. أداة صندوق التحقق CheckBox.
9. أداة الزر Button.

## مقدمة:

- تطورت لغات البرمجة بشكل لافت للنظر في الآونة الأخيرة وأصبحت هناك إمكانيات عالية جداً في لغات البرمجة بحيث تقدم للمبرمجين كل ما يحتاجونه في أبسط وأسهل وأسرع شكل ممكن ، وقد انعكس هذا التقدم علي تصميم واجهة أي برنامج ، حيث أصبحت عملية تصميم الواجهة والتحكم في خصائصها عملية غاية في السهولة.
- وكون لغة Visual Basic من اللغات السهلة جداً ، فإنها تقدم للمبرمجين مجموعة كبيرة جداً من الأدوات التي تساعد في بناء الواجهة بطريقة جذابة.
- وفي هذا الفصل نتعرف علي أهم الأدوات المستخدمة في تصميم الواجهة ثم نتعرف في الفصل القادم علي كيفية معالجة الأحداث Events Handling التي تتيح للمستخدم التفاعل مع البرنامج كما سنري في الفصل القادم.
- سوف نتعرف في هذا الفصل علي الأدوات التالية:
  1. أداة النموذج Form.
  2. أداة العنوان Label.
  3. أداة النص TextBox.
  4. أداة قائمة الاختيارات ComboBox.
  5. أداة زر الراديو RadioButton.
  6. أداة صندوق التحقق CheckBox.
  7. أداة الزر Button.

## إنشاء الواجهة الرسومية GUI:

- إن عملية إدخال الأدوات في أي تطبيق تنشئه هي عملية مكررة ومتشابهة ، حيث توفر لنا لغة Visual Basic شريط أدوات Toolbar يحتوي علي

جميع الأدوات المتاحة ، ويجب علي المبرمج اختيار الأداة التي يريدها ثـم يقوم برسمها في نافذة التطبيق.

- وبعد إدخال الأداة المطلوبة ، فإنه يتم تحديد خصائص الأداة من خلال شاشة الخصائص Properties Window ، فمثلاً يمكنك تغيير طـول وعـرض النموذج Form ، كما يمكن تغيير شكل ولون وحجم الخط... إلخ.
- تختلف خصائص كل أداة عن الأخري علي الرغم من وجود تشـابه بـين بعض الخصائص ولكن لن تختلف طريقة التحكم في هذه الخصائص من أداة لأخري.
- إذن نلخص الفقرة السابقة كالآتي:
- لإنشاء أي أداة فإننا نختار هذه الأداة من شريط الأدوات Toolbar الخاص بإدخال جميع الأدوات ثم نقوم بتحديد خصائصها من خلال شاشة الخصائص Properties Window وبذلك نكون قد انتهينا من إدخال أي أداة نريـدها بمنتهي السهولة.
- إذن كل ما نحتاج إلي معرفته هو اسم الأداة وأهميتها ومجال استخدامها مع معرفة أهم خصائصها حتي نستطيع إظهار الأداة بالشكل المطلوب.
- سوف نبدأ بتوضيح أول أداة وهي أداة النموذج Form.

## أولاً: أداة النموذج Form:

أداة النموذج Form هي المسئولة عن إظهار نافذة البرنامج بكل ما تحتويه مـن أدوات تتيح للمستخدم التفاعل مع البرنامج.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الخاصية | الاستخدام |
|---|---|
| لون الخلفية BackColor | تغيير لون خلفية النموذج Form. |
| صورة الخلفية BackgroundImage | تغيير صورة خلفية النموذج Form. |
| المؤشر Cursor | تغيير شكل مؤشر الفارة Mouse. |
| العنوان Text | تغيير عنوان النافذة. |
| الاسم (Name) | تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. |
| الحجم Size | تحديد طول وعرض النموذج Form. |
| الموضع الابتدائي StartPosition | تحديد الموضع الذي ستظهر فيه النافذة عند تشغيل التطبيق فيمكن مثلاً تحديد موضع ظهورها ليكون في منتصف الشاشة بالضبط. |
| الأيقونة Icon | تحديد الأيقونة التي ستظهر في أعلي يسار النافذة بجانب عنوان النافذة. |

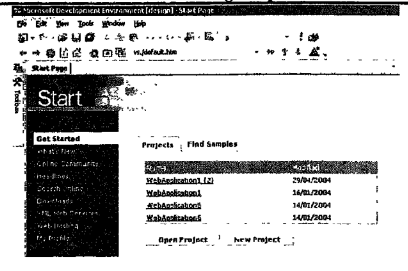
سنقوم في المثال التالي بتوضيح كيفية إنشاء أداة النموذج Form.

## خطوات التنفيذ:

1. افتح البرنامج عن طريق اختيار

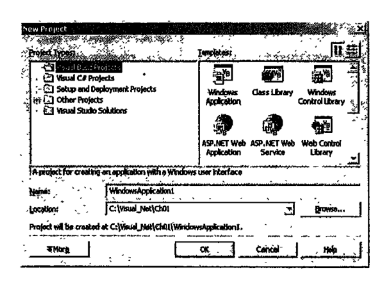
Start → Programs → Microsoft Visual Studio.Net → Microsoft Visual Studio.Net

لتظهر لك نافذة البرنامج كما هو واضح في شكل 1.



شكل (1 أداة النموذج Form الخطوة الأولي)

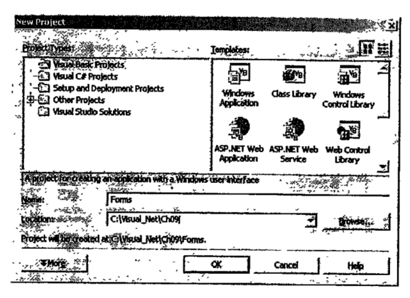
2. لإنشاء مشروع Project جديد فيمكنك الضغط على الزر المسمى New Project مباشرة أو عن طريق القوائم بفتح القائمة File ثم New ثم Project أو باستخدام لوحة المفاتيح Keyboard بالضغط على الأزرار Ctrl + Shift + N لتظهر لنا الشاشة كما هو واضح في شكل 2.



شكل (2 أداة النموذج Form الخطوة الثانية)

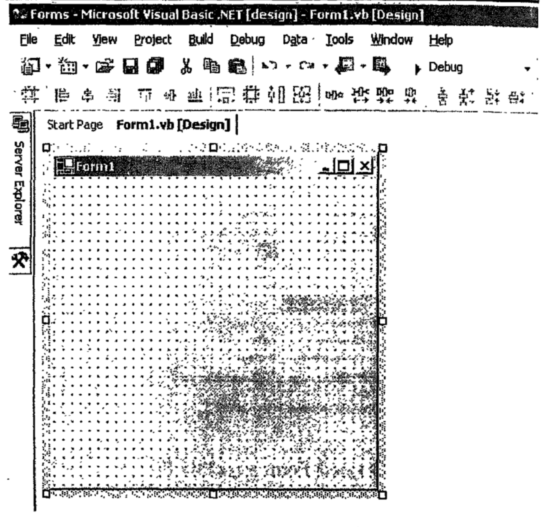
3. من الناحية اليسرى سنختار نوع المشروع "Project Types" الذي نريد إنشاؤه وبالطبع سنختار نوع المشروع ليكون " Visual Basic Projects".
4. ومن الناحية اليمنى سنختار قالب المشروع "Templates" ليكون "Windows Application" لأننا نريد أن ننشيء برنامج تطبيق نوافذ.
5. في خانة الاسم "Name" نحدد اسماً للمشروع وليكن Forms وفي خانة المكان Location نحدد مسار المشروع وليكن "C:\Visual_Net\Ch09" ويمكنك بالطبع اختيار أي مسار آخر.

شكل 3 يبين الاختيارات النهائية للمشروع ويمكنك الضغط على الزر "OK" لبدء تنفيذ المشروع.



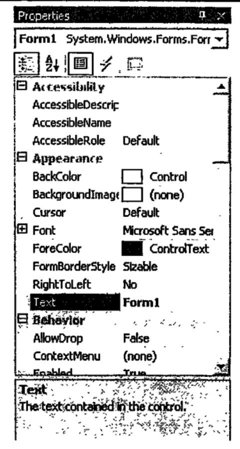

شكل 3) أداة النموذج Form الخطوة الخامسة)

6. شكل 4 يوضح نافذة التطبيق التي أنشأتها لغة Visual Basic.




شكل (4 أداة النموذج Form الخطوة السادسة)

7. نلاحظ هنا أن لغة Visual Basic قامت بإنشاء نموذج Form بشكل تلقائي وكل ما نحتاجه الآن هو التعديل في خصائص هذا النموذج Form لتحقق الشكل الذي نريده.
8. يتم إظهار خصائص أي أداة عن طريق الضغط بالزر الأيمن للفارة Mouse على الأداة ثم اختيار Properties ليتم إظهار شاشة الخصائص Properties Window كما هو واضح في شكل 5.




(شكل 5) أداة النموذج Form الخطوة الثامنة

9. يتضح لنا من شكل 5 ظهور جميع خصائص النموذج Form والتي قمنا بتوضيح بعضها في الجدول السابق.

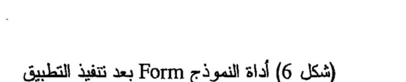
10. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجودة في عمود "القيمة الجديدة". (في جميع الخصائص التالية قم بالبحث عن اسم الخاصية في العمود الأول من شاشة الخصائص Properties Window الموضحة في شكل 5 السابق ثم قم بمسح القيمة المناظرة لكل خاصية من العمود الثاني ثم قم بتعديلها بالقيم الموجودة في الجدول التالي).

| اسم الخاصية | القيمة الجديدة |
|---|---|
| لون الخلفية BackColor | 224; 224; 224 |
| المؤشر Cursor | Hand |
| العنوان Text | My Data |
| الاسم (Name) | myData |
| الحجم Size | 800;600 |
| الموضع الابتدائي StartPosition | CenterScreen |

ملحوظة:

في معظم الخصائص ستجد غالباً سهماً للاختيارات عند الضغط في العمود الثاني من شاشة الخصائص Properties Window بحيث يمكننا اختيار القيمة الجديدة من سهم الاختيارات بدلاً من كتابة القيمة الجديدة بأنفسنا.

11. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 6.




(شكل 6) أداة النموذج Form بعد تنفيذ التطبيق

12. نلاحظ تغيير لون الخلفية وعنوان النافذة بالإضافة لتغيير باقي الخصائص السابق توضيحها في الجدول السابق وبذلك نكون قد انتهينا من تنفيذ أداة النموذج Form وسنقوم في الأمثلة القادمة بتوضيح كيفية إضافة باقي الأدوات إلي هذا النموذج Form.

## ثانياً: أداة العنوان Label:

تستخدم هذه الأداة لوضع صورة أو نص ثابت في التطبيق بحيث لا يكون بإمكان المستخدم التعديل في هذا النص أو التفاعل معه أثناء عمل البرنامج ، أي أن أداة العنوان Label هي أداة للقراءة فقط وتستخدم لإعطاء معلومة للمستخدم عن البيانات المطلوب إدخالها.

فمثلاً إذا أنشأت تطبيقاً يطلب من المستخدم إدخال اسمه ، إذن فأنت تحتاج لوضع كلمة الاسم كنص ثابت في التطبيق لبيان أنك تريد من المستخدم إدخال اسمه ولا يحتاج المستخدم أن يعدل في هذا النص.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

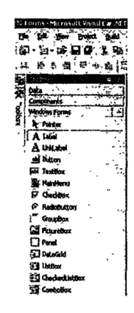
| اسم الخاصية | الاستخدام |
|---|---|
| الحجم التلقائي AutoSize | تغيير حجم أداة العنوان Label حتي يحتوي النص الذي سيظهر علي الأداة. |
| شكل الخط Font | تغيير شكل الخط Font للنص الذي سيظهر علي الأداة. |
| لون الخط ForeColor | تغيير لون الخط Font للنص الذي سيظهر علي الأداة. |
| الاسم (Name) | تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. |

| الموضع Location | تحديد موقع أداة العنوان Label في نافذة التطبيق. |
|---|---|
| الحجم Size | تحديد حجم أداة العنوان Label. |
| النص Text | تحديد النص الذي سيظهر علي أداة العنوان Label. |
| محاذاة النص TextAlign | تحديد محاذاة النص الذي سيظهر علي أداة العنوان Label. |

سنقوم في المثال التالي بتوضيح كيفية إضافة أداة العنوان Label إلي نافذة التطبيق.

## خطوات التنفيذ:

1. قم بالضغط علي صندوق الأدوات Toolbox ثم اضغط علي أداة العنوان Label كما هو واضح في شكل 7.




(شكل 7) صندوق الأدوات Toolbox

2. قم بتحريك مؤشر الفارة Mouse إلي النموذج Form ثم اضغط في المكان الذي تريد وضع أداة العنوان Label فيه وتأكد أن شكل النموذج Form أصبح كما هو واضح في شكل 8.

شكل 8) أداة العنوان Label)

3. نلاحظ الآن أن شاشة الخصائص Properties Window قد تغيرت لتبين خصائص الأداة الجديدة ، وإذا قمت بالضغط في أي مكان خطأ بحيث اختفت خصائص أداة العنوان Label ، فيمكنك الضغط بالزر الأيمن للفارة Mouse على أداة العنوان Label ثم اختيار Properties.
4. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجوذة في عمود "القيمة الجديدة".

| اسم الخاصية | القيمة الجديدة |
|---|---|
| الحجم التلقائي AutoSize | True |
| شكل الخط Font | Pal- tino Linotype; 9.75pt |
| لون الخط ForeColor | Blue |
| الموضع Location | 56;40 |
| النص Text | Name: |

5. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 9.

My Data

Name:

(شكل 9) أداة العنوان Label بعد تنفيذ التطبيق

## ثالثاً: أداة النص TextBox:

تسمح أداة النص TextBox للمستخدم بإدخال حروف من لوحة المفاتيح Keyboard ، ولذلك تعتبر هذه الأداة من الأدوات الرئيسية التي تتيح للمستخدم إدخال معلومات معينة للتطبيق فمثلاً قد تريد من المستخدم إدخال اسمه ، ولذلك نحتاج إلي أداة النص TextBox لكي تسمح للمستخدم بأن يقوم بإدخال البيانات التي يريدها.

يمكنك من خلال كود البرمجة تحديد ما إذا كانت أداة النص TextBox تسمح للمستخدم بالكتابة في سطر واحد فقط أو في أكثر من سطر من خلال خاصية تعدد الأسطر MultiLine كما سنري لاحقاً.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

| الاستخدام | اسم الخاصية |
|---|---|
| تغيير شكل الخط Font للنص الذي سيتم كتابته في الأداة. | شكل الخط Font |
| تغيير لون الخط Font للنص الذي سيتم كتابته في الأداة. | لون الخط ForeColor |
| تغيير لون الخلفية للنص الذي سيتم كتابته في الأداة. | لون الخلفية BackColor |
| تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. | الاسم (Name) |
| تحديد موقع أداة النص TextBox في نافذة التطبيق. | الموضع Location |
| تحديد حجم أداة النص TextBox. | الحجم Size |
| تحديد النص الافتراضي الذي سيظهر في أداة النص TextBox. | النص Text |
| تحديد محاذاة النص الذي سيتم كتابته في الأداة. | محاذاة النص TextAlign |
| تحديد أقصي عدد من الحروف للنض الممكن إدخاله في الأداة. | أقصي غدد من الحروف MaxLength |
| تحديد ما إذا كانت أداة النص Textbox تسمح بالكتابة علي عدة أسطر أم لا. | تعدد الأسطر MultiLine |
| تحديد ما إذا كانت هناك أشركة تمرير ScrollBars ظاهرة في الأداة أم لا. | أشرطة التمرير ScrollBars |

سنقوم في المثال التالي بتوضيح كيفية إضافة أداة النص TextBox إلي نافذة التطبيق.

**خطوات التنفيذ:**

1. قم بالضغط علي صندوق الأدوات Toolbox ثم اضغط علي أداة النص TextBox كما هو واضح في شكل 7 السابق.
2. قم بتحريك مؤشر الفارة Mouse إلي النموذج Form ثم اضغط في المكان الذي تريد وضع أداة النص TextBox فيه وتأكد أن شكل النموذج Form أصبح كما هو واضح في شكل 10.

My Data

Name: textBox1

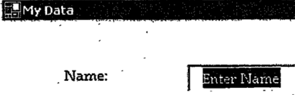
(شكل 10) أداة النص TextBox

3. تأكد الآن أن شاشة الخصائص Properties Window قد تغيرت لتبين خصائص الأداة الجديدة.
4. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجودة في عمود "القيمة الجديدة".

| اسم الخاصية | القيمة الجديدة |
|---|---|
| شكل الخط Font | .Palatino Linotype; 9.75pt |

| لون الخط ForeColor | Blue |
|---|---|
| لون الخلفية BackColor | 255;255;192 |
| الاسم (Name) | myName |
| الموضع Location | 170;40 |
| الحجم Size | 100;25 |
| النص Text | Enter Name |
| محاذاة النص TextAlign | Center |
| أقصي عدد من الحروف MaxLength | 15 |

5. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 11.

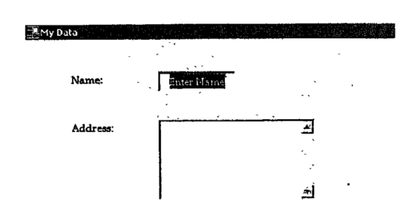



(شكل 11) أداة النص TextBox بعد تنفيذ التطبيق

6. قم بإضافة أداة عنوان Label جديدة ثم قم بتغيير خاصية النص Text لتصبح "Address:" ثم قم بإضافة أداة نص TextBox جديدة ثم قم بتحديد خصائصها كما هو واضح في الجدول التالي.

| القيمة الجديدة | اسم الخاصية |
|---|---|
| myAddress | (Name) الاسم |
| Both | ScrollBars أشرطة التمرير |
| لا تكتب شيئاً هنا واتركها خالية | Text النص |
| true | MultiLine تعدد الأسطر |
| 200;100 | Size الحجم |
| 170;100 | Location الموضع |

7. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 12.




(شكل 12) أداة النص TextBox بعد تنفيذ التطبيق

## رابعاً: أداة قائمة الاختيارات ComboBox:

تتيح لنا أداة قائمة الاختيارات ComboBox اختيار قيمة واحدة فقط من عدة اختيارات متاحة ، فمثلاً إذا كنت تريد من المستخدم إدخال الدولة التي ولد فيها ،

فتستطيع إنشاء قائمة اختيارات تحتوي علي جميع أسماء الدول وعلي المستخدم أن يختار قيمة واحدة فقط من الاختيارات المتاحة وليس من حقه اختيار أي قيمة أخري غير موجودة في القائمة ، أي أن المستخدم محدود بالقيم التي نحددها في قائمة الاختيارات.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

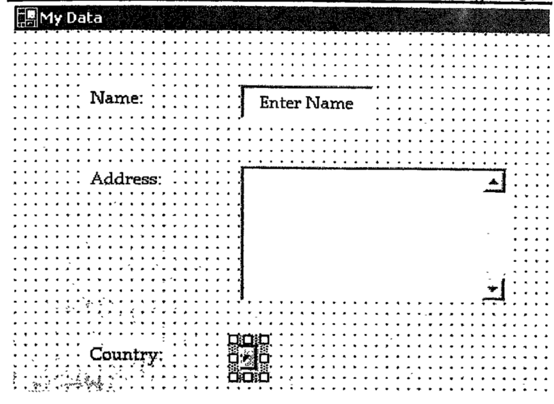
| اسم الخاصية | الاستخدام |
|---|---|
| شكل الخط Font | تغيير شكل الخط Font للنص الذي سيتم كتابته في الأداة. |
| لون الخط ForeColor | تغيير لون الخط Font للنص الذي سيتم كتابته في الأداة. |
| لون الخلفية BackColor | تغيير لون الخلفية للنص الذي سيتم كتابته في الأداة. |
| الاسم (Name) | تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. |
| الموضع Location | تحديد موقع أداة قائمة الاختيارات ComboBox في نافذة التطبيق. |
| الحجم Size | تحديد حجم أداة قائمة الاختيارات ComboBox. |
| النص Text | تحديد النص الافتراضي الذي سيظهر في الأداة. |
| أقصي عدد للعناصر في قائمة الاختيارات | تحديد أقصي عدد من العناصر يتم إظهاره في قائمة الاختيارات بدون ظهور أشرطة التمرير |

| | |
|---|---|
| MaxDropDownItems | .ScrollBars |
| أقصي عدد من الحروف MaxLength | أقصي عدد من الحروف للنص الممكن إدخاله في الأداة. |
| الترتيب Sorted | تحديد ما إذا كانت العناصر في الأداة ستظهر مرتبة أبجدياً أم لا. |
| العناصر Items | تحديد قائمة الاختيارات في الأداة. |

سنقوم في المثال التالي بتوضيح كيفية إضافة أداة قائمة الاختيارات ComboBox إلي نافذة التطبيق.

## خطوات التنفيذ:

1. قم بإضافة أداة عنوان Label جديدة ثم قم بتغيير خاصية النص Text لتصبح ":Country".
2. قم بالضغط علي صندوق الأدوات Toolbox ثم اضغط علي أداة قائمة الاختيارات ComboBox كما هو واضح في شكل 7 السابق.
3. قم بتحريك مؤشر الفارة Mouse إلي النموذج Form ثم اضغط في المكان الذي تريد وضع أداة قائمة الاختيارات ComboBox فيه وتأكد أن شكل النموذج Form أصبح كما هو واضح في شكل 13.




(شكل 13) أداة قائمة الاختيارات ComboBox

4. تأكد الآن أن شاشة الخصائص Properties Window قد تغيرت لتبين خصائص الأداة الجديدة.
5. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجودة في عمود "القيمة الجديدة"..

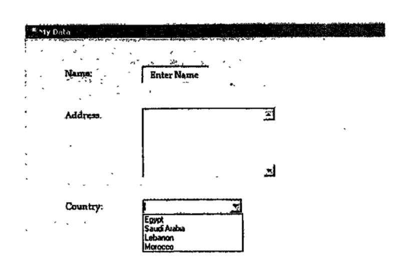
| اسم الخاصية | القيمة الجديدة |
|---|---|
| (Name) الاسم | myCountry |
| Location الموضع | 170;230 |
| Size الحجم | 150;21 |
| Text النص | لا تكتب شيئاً هنا واتركها خالية |

| | |
|---|---|
| Egypt<br>Saudi Arabia<br>Lebanon<br>Morocco | Items العناصر |

ملحوظة:

عند الضغط في العمود الثاني المناظر لخاصية العناصر Items ، فإنه يتم إظهار زر موجود عليه ثلاث نقاط ، وعند الضغط علي هذا الزر ، فإنه يتم فتح شاشة لكتابة قائمة الاختيارات فيها ، ويمكنك كتابة الاختيارات كما سبق أن أوضحنا في الجدول السابق بشرط كتابة كل اختيار في سطر منفصل (أي يجب الضغط علي زر الإدخال Enter بعد كتابة كل اختيار).

6. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 14.




(شكل 14) أداة قائمة الاختيارات ComboBox بعد تنفيذ التطبيق

## خامساً: أداة زر الراديو RadioButton:

تستخدم أداة زر الراديو RadioButton عندما نريد أن نختار قيمة واحدة فقط من عدة اختيارات متاحة ، وبذلك فهي تتشابه مع أداة علبة الفرقة ComboBox ولكن الاختلاف يتمثل أساساً في الحجم الذي تحتاجه لكتابة الاختيارات في نافذة التطبيق ، فأداة علبة الفرقة ComboBox تحتاج حجم أقل من أداة زر الراديو RadioButton لإظهار جميع الاختيارات المتاحة للمستخدم ، ولذلك لا تستخدم أداة زر الراديو RadioButton إلا إذا كانت قائمة الاختيارات محدودة جداً وإلا احتجت إلي مساحة كبيرة جداً لإظهار جميع الاختيارات التي تريدها.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الخاصية | الاستخدام |
|---|---|
| شكل الخط Font | تغيير شكل الخط Font للنص الذي سيتم كتابته بجانب الأداة. |
| لون الخط ForeColor | تغيير لون الخط Font للنص الذي سيتم كتابته بجانب الأداة. |
| لون الخلفية BackColor | تغيير لون الخلفية للنص الذي سيتم كتابته بجانب الأداة. |
| الاسم (Name) | تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. |
| الموضع Location | تحديد موقع أداة زر الراديو RadioButton في نافذة التطبيق. |
| الحجم Size | تحديد حجم أداة زر الراديو RadioButton. |
| النص Text | تحديد النص الذي سيظهر بجانب الأداة. |

| محاذاة العلامة CheckAlign | تحديد محاذاة علامة زر الراديو .RadioButton |
|---|---|
| مختارة Checked | تحديد ما إذا تم اختيار الأداة أم لا. |

سنقوم في المثال التالي بتوضيح كيفية إضافة أداة زر الراديو RadioButton إلي نافذة التطبيق.

## خطوات التنفيذ:

1. قم بإضافة أداة عنوان Label جديدة ثم قم بتغيير خاصية النص Text لتصبح ".Marital Status:".
2. قم بالضغط علي صندوق الأدوات Toolbox ثم اضغط علي أداة زر الراديو RadioButton كما هو واضح في شكل 7 السابق.
3. قم بتحريك مؤشر الفارة Mouse إلي النموذج Form ثم اضغط في المكان الذي تريد وضع أداة زر الراديو RadioButton فيه وتأكد أن شكل النموذج Form أصبح كما هو واضح في شكل 15.

(شكل 15) أداة زر الراديو RadioButton

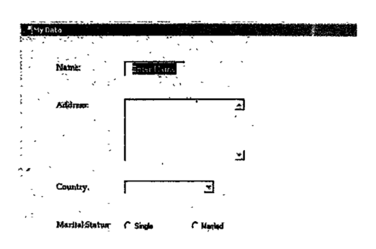
4. تأكد الآن أن شاشة الخصائص Properties Window قد تغيرت لتبين خصائص الأداة الجديدة.
5. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجودة في عمود "القيمة الجديدة".

| اسم الخاصية | القيمة الجديدة |
|---|---|
| (Name) الاسم | single1 |
| Location الموضع | 170;290 |
| Size الحجم | 60;24 |
| Text النص | Single |

6. قم بإضافة أداة زر راديو RadioButton أخرى بجانب الأداة الأولـــي ثم قم بالتعديل في الخصائص الموضحة فـــي الجـــدول التـــالي بـــالقيم الموجودة في عمود "القيمة الجديدة".

| اسم الخاصية | القيمة الجديدة |
|---|---|
| (Name) الاسم | married |
| Location الموضع | 280; 290 |
| Size الحجم | 65; 24 |
| Text النص | Married |

7. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشـــكل كما هو واضح في شكل 16.




(شكل 16) أداة زر الراديو RadioButton بعد تنفيذ التطبيق

## سادساً: أداة صندوق التحقق CheckBox:

تمكننا أداة صندوق التحقق CheckBox من اختيار قيمة من اثنين إما نعم true وإما لا false ، فمثلاً قد تريد من المستخدم بيان ما إذا كان عنده أطفـال أم لا ، إذن هنا اختيار المستخدم إما أن عنده أطفال أو لا ، أو بمعني أخـر: نعـم أو لا وهنا تظهر فائدة أداة صندوق التحقق CheckBox.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

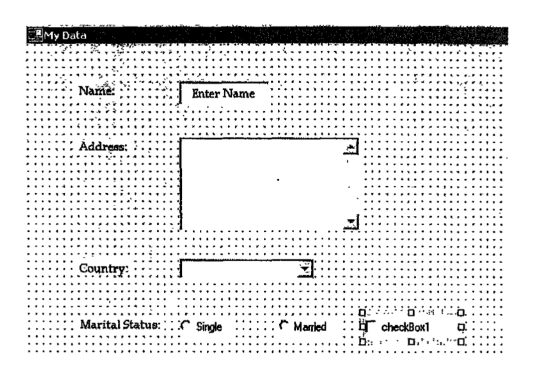
| اسم الخاصية | الاستخدام |
| --- | --- |
| شكل الخط Font | تغيير شكل الخط Font للنص الذي سيتم كتابته بجانب الأداة. |
| لون الخط ForeColor | تغيير لون الخط Font للنص الذي سيتم كتابته بجانب الأداة. |
| لون الخلفية BackColor | تغيير لون الخلفية للنص الذي سيتم كتابته بجانب الأداة. |
| الاسم (Name) | تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. |
| الموضع Location | تحديد موقع أداة صندوق التحقق CheckBox في نافذة التطبيق. |
| الحجم Size | تحديد حجم أداة صندوق التحقق CheckBox. |
| النص Text | تحديد النص الذي سيظهر بجانب الأداة. |
| محاذاة العلامة CheckAlign | تحديد محاذاة علامة صندوق التحقق CheckBox. |

| مختارة Checked | تحديد ما إذا تم اختيار الأداة أم لا. |
|---|---|

سنقوم في المثال التالي بتوضيح كيفية إضافة أداة صندوق التحقق CheckBox إلي نافذة التطبيق.

**خطوات التنفيذ:**

1. قم بالضغط علي صندوق الأدوات Toolbox ثم اضغط علي أداة صندوق التحقق CheckBox كما هو واضح في شكل 7 السابق.
2. قم بتحريك مؤشر الفارة Mouse إلي النموذج Form ثم اضغط في المكان الذي تريد وضع أداة صندوق التحقق CheckBox فيه وتأكد أن شكل النموذج Form أصبح كما هو واضح في شكل 17.




(شكل 17) أداة صندوق التحقق CheckBox

3. تأكد الآن أن شاشة الخصائص Properties Window قد تغيرت لتبين خصائص الأداة الجديدة.
4. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجودة في عمود "القيمة الجديدة".

| اسم الخاصية | القيمة الجديدة |
|---|---|
| الاسم (Name) | children |
| الموضع Location | 390;290 |
| الحجم Size | 100;24 |
| النص Text | Have Children |

5. قم بإضافة أداة عنوان Label جديدة ثم قم بتغيير خاصية النص Text لتصبح "Number of Children:" ثم قم بإضافة أداة نص TextBox جديدة ثم قم بتغيير خاصية الاسم (Name) لتصبح "number". (انظر شكل 18).
6. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 18.

My Data

Name. Enter Name

Address:

Country:

Marital Status: Single Married Have Children

Number of Children

(شكل 18) أداة صندوق التحقق CheckBox بعد تنفيذ التطبيق

## سابعاً: أداة الزر Button:

تستخدم أداة الزر Button لإظهار أزرار فى نافذة التطبيق حيث يستخدم الــزر في تنفيذ أوامر معينة عندما يضغط المستخدم علي الزر Button.

تتوافر العديد من الخصائص لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الخاصية | الاستخدام |
|---|---|
| شكل الخط Font | تغيير شكل الخط Font للنص الذي سيتم كتابته علي الأداة. |
| لون الخط ForeColor | تغيير لون الخط Font للنص الذي سيتم كتابته |

| | |
|---|---|
| | علي الأداة. |
| لون الخلفية BackColor | تغيير لون الخلفية للنص الذي سيتم كتابته علي الأداة. |
| الاسم (Name) | تحديد اسم المتغير الذي يستخدم في تحديد خصائص الأداة من خلال كود البرمجة. |
| الموضع Location | تحديد موقع أداة الزر Button في نافذة التطبيق. |
| الحجم Size | تحديد حجم أداة الزر Button. |
| النص Text | تحديد النص الذي سيظهر علي الأداة. |

سنقوم في المثال التالي بتوضيح كيفية إضافة أداة صندوق الـزر Button إلـي نافذة التطبيق.

## خطوات التنفيذ:

1. قم بالضغط علي صندوق الأدوات Toolbox ثم اضغط علي أداة الزر Button كما هو واضح في شكل 7 السابق.
2. قم بتحريك مؤشر الفارة Mouse إلي النموذج Form ثم اضغط فـي المكان الذي تريد وضع أداة الزر Button فيه وتأكد أن شكل النموذج Form أصبح كما هو واضح في شكل 19.

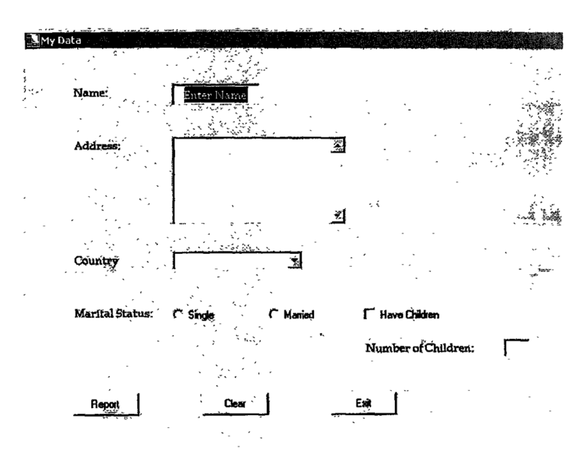
(شكل 19) أداة الزر Button

3. تأكد الآن أن شاشة الخصائص Properties Window قد تغيرت لتبين خصائص الأداة الجديدة.

4. يمكنك الآن التعديل في الخصائص الموضحة في الجدول التالي بالقيم الموجودة في عمود "القيمة الجديدة".

| اسم الخاصية | القيمة الجديدة |
|---|---|
| (Name) الاسم | report |
| Location الموضع | 56;390 |
| Size الحجم | 75;25 |
| Text النص | Report |

5. قم بإضافة أداة زر Button جديدة ثم قم بتغيير خاصية النص Text لتصبح "Clear" وخاصية الاسم (Name) لتكون clear ، ثم قم بإضافة أداة زر Button أخري ثم قم بتغيير خاصية النص Text لتصبح "Exit" وخاصية الاسم (Name) لتكون quit.
6. نحتاج الآن لتنفيذ التطبيق لرؤية نتيجة تغيير الخصائص السابقة ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 لتحصل على الشكل كما هو واضح في شكل 20.




(شكل 20) أداة الزر Button بعد تنفيذ التطبيق

# الفصل العاشر

## معالجة الأحداث

## Event Handling

في هذا الفصل سوف نتعرف علي كيفية معالجة الأحداث Events Handling في لغة Visual Basic والتي تمكننا من التفاعل مع المستخدم وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. الحدث Event.
3. معالجة الأحداث Events Handling.
4. أحداث أداة النموذج Form.
5. أحداث أداة العنوان Label.
6. أحداث أداة النص TextBox.
7. أحداث أداة قائمة الاختيارات ComboBox.
8. أحداث أداة زر الراديو RadioButton.
9. أحداث أداة صندوق التحقق CheckBox.
10. أحداث أداة الزر Button.
11. الكود الكامل.

معالجة الأحداث
Event Handling

## مقدمة:

- تعلمنا في الفصل السابق كيفية بناء واجهة البرنامج عن طريق استخدام مجموعة من الأدوات الجاهزة في اللغة.
- ولكن هناك المزيد ، لأن تصميم وتنفيذ شاشات الواجهة الرسومية GUI فقط يعتبر كإنشاء سيف بلا مقبض ، فلا بد للبرنامج أن يتفاعل مع المستخدم وأن تكون هناك قابلية لتبادل المعلومات بين المستخدم والبرنامج بحيث يمكن للمستخدم أن يدخل بعض البيانات للبرنامج ثم يعطي للبرنامج أمراً لتنفيذ وإخراج نتيجة ما تعتمد علي البيانات التي أدخلها المستخدم.
- وفي هذا الفصل نتعرف علي أهم الأدوات المستخدمة في معالجة الأحداث Events Handling التي تتيح للمستخدم التفاعل مع البرنامج كما سنري في هذا الفصل.
- قبل البدء في شرح الأمثلة ، فلا بد أولاً أن نتعرف علي بعض المفاهيم الأساسية.

## الحدث Event:

- الحدث Event هو أى شئ يفعله المستخدم في نافذة التطبيق ، فمثلاً إذا ضغط المستخدم علي زر Button في التطبيق فإنه بذلك يكون قد أنشأ حدثاً Event ، وبالمثل يتم إنشاء حدث Event عند الضغط علي زر الفارة Mouse وعند تحريك الفارة Mouse وعند فتح نافذة التطبيق نفسها أو إغلاقها أو تصـ.ـنيرها Minimize أو تكبيرها Maximize... إلخ.
- أي أنه توجد العديد من الأحداث Events في أي برنامج ولكن ليست كل أفعال المستخدم ذات أهمية ، فنجد أن بعض الأحداث Events ذات أهمية

ولا بد أن يكون للبرنامج رد فعل لهذه الأحداث Events أما بعض الأحداث Events الأخري فلا أهمية لها ولا نريد إنشاء رد فعل لها.

- ولكي نفهم النقطة السابقة فيجب أن نذكر أن أهمية الحدث Event تنبع من احتياجات التطبيق نفسه ، فمثلاً إذا قمت بتصميم برنامج رسم ، إذن فتحريك وسحب وضغط مؤشر الفارة Mouse تعتبر أحداثاً Events ذات أهمية قصوي لأن معظم برامج الرسم تعتمد اعتماداً كبيراً علي الفارة Mouse ، أما إذا قمت بتصميم برنامج كتابة ، إذن فتحريك مؤشر الفارة Mouse يعتبر حدثاً Event بلا أهمية ، فتجد أن برنامج الوورد Word مثلاً لا يقوم بتنفيذ أي وظيفة عند تحريك مؤشر الفارة Mouse ، ولكن الضغط علي زر الفارة Mouse يعتبر حدثاً Event ذا أهمية قصوي لأنه يحدد موضع بداية الكتابة وبالمثل بالنسبة لسحب مؤشر الفارة Mouse لأنه يقوم بتظليل select النصوص في البرنامج.
- إذن نخرج من المناقشة السابقة بأن كل أداة لها أحداث Event خاصة بها بحيث أن بعض الأحداث Event تكون ذات أهمية ونريد أن يكون للبرنامج رد فعل لهذه الأحداث Events.
- ويعتمد رد فعل البرنامج علي الأوامر التي نكتبها لكل حدث Event وهو ما يعرف باسم معالجة الحدث Event Handling.

## معالجة الأحداث Events Handling:

- هى رد فعل البرنامج للحدث Event الذى يطلقه المستخدم وتتمثل في كتابة مجموعة من الأوامر لتقوم بتنفيذ وظيفة معينة عند وقوع الحدث Event.
- وتعتبر عملية معالجة الأحداث Handling Events في أي تطبيق تنشئه هي عملية مكررة ومتشابهة (تماماً مثل إنشاء الواجهة الرسومية GUI) ،

حيث توفر لنا لغة Visual Basic شاشة لأحداث Events كل أداة من الأدوات المتاحة ، فمثلاً يمكنك إظهار رسالة للمستخدم إذا ضغط علي زر Button معين ، كما يمكنك تغيير لون وخلفية النموذج Form عندما يتحرك مؤشر الفارة Mouse علي النموذج Form ... إلخ.

- تختلف أحداث Events كل أداة عن الأخري علي الرغم من وجود تشابه بين بعض الأحداث Events ولكن لن تختلف طريقة إنشاء أوامر معالجة الحدث Event Handling لهذه الأحداث Events من أداة لأخري.
- إذن نلخص الفقرة السابقة كالآتي:
- لإنشاء أوامر لأحداث Events أي أداة ، فإننا نقوم بإظهار أحداث Events هذه الأداة ثم نقوم بتحديد الحدث Event المطلوب معالجته ثم نقوم بكتابة الأوامر التي نريد تنفيذها عند وقوع هذا الحدث Event.
- إذن كل ما نحتاج إلي معرفته هو أسماء أحداث Events كل أداة وأهميتها ومجال استخدامها مع إعطاء أمثلة للأوامر التي يمكن تنفيذها حتي نستطيع تنفيذ الأوامر بالشكل المطلوب.
- سوف نبدأ بتوضيح أحداث Events أول أداة وهي أداة النموذج Form.

## أولاً: أحداث أداة النموذج Form Events:

تتوافر الأحداث Events لهذه الأداة كما هو واضح في الجدول التالي.

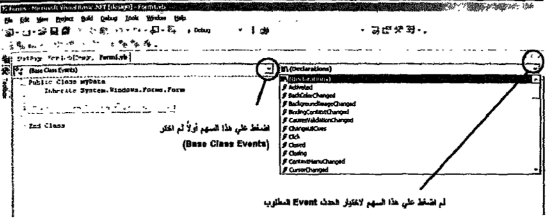
| اسم الحدث Event | وقت الحدوث |
|---|---|
| Load | عند فتح النموذج Form. |
| MouseDown | عندما يضغط المستخدم علي زر الفارة Mouse داخل مساحة النموذج Form. |

| | |
|---|---|
| MouseEnter | عندما يدخل مؤشر الفارة Mouse داخل مساحة النموذج Form. |
| MouseHover | عندما يقف المستخدم بمؤشر الفارة Mouse لمدة ثواني داخل مساحة النموذج Form. |
| MouseLeave | عندما يخرج مؤشر الفارة Mouse من مساحة النموذج Form. |
| MouseMove | عندما يحرك المستخدم مؤشر الفارة Mouse داخل مساحة النموذج Form. |
| MouseUp | عندما يحرر المستخدم زر الفارة Mouse بعد الضغط عليه داخل مساحة النموذج Form. |

سنقوم في المثال التالي بتوضيح كيفية استخدام أحداث Events هذه الأداة.

## خطوات التنفيذ:

1. تأكد من فتح المشروع السابق إنشائه في الفصل السابق.
2. تأكد من إظهار خصائص النموذج Form عن طريق الضغط بالزر الأيمن للفارة Mouse علي النموذج Form ثم اختيار Properties ليتم إظهار شاشة الخصائص Properties Window ثم افتح قائمة View ثم Code أو اضغط علي الزر F7 ليتم فتح كود الأوامر والذي نقوم فيه بإظهار الأحداث Events عن الطريق الضغط علي سهم الاختيارات الموضح في شكل 1 ثم اختيار (Base Class Events) ثم الضغط علي سهم الاختيارات الموضح في شكل 1 ليتم عرض الأحداث Events المختلفة.

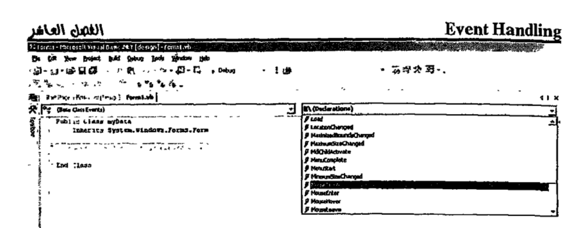



شكل 1) أحداث النموذج Form Events الخطوة الثانية

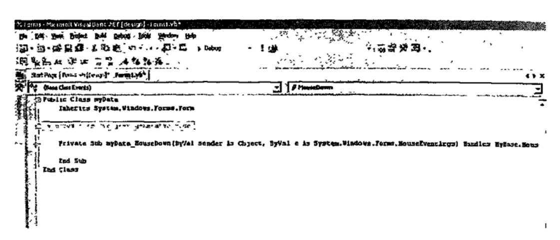
ملحوظة:

يطلق علي سهم الاختيارات الأيسر اسم "اسم الفصيلة Class Name" وعلي سهم الاختيارات الأيمن اسم "اسم الدالة Method Name" وسنقوم باستخدام هذه المسميات للإشارة لهذه الأسهم في جميع الأمثلة القادمة.

3. يتضح لنا من شكل 1 ظهور جميع أحداث النموذج Form Events والتي قمنا بتوضيح بعضها في الجدول السابق.
4. نريد الآن تغيير لون خلفية النموذج BackColor عندما يضغط المستخدم علي زر الفارة Mouse وهذا يعني أننا نريد إنشاء رد فعل للحدث MouseDown ولذلك قم بالبحث عن الحدث MouseDown كما هو واضح في شكل 2 لتتغير الشاشة كما هو واضح في شكل 3.



(شكل 2) أحداث النموذج Form Events الخطوة الرابعة



(شكل 3) أحداث النموذج Form Events الخطوة الرابعة

5. نلاحظ الآن أن لغة Visual Basic أنشأت دالة Subroutine جديدة وتلاحظ أن مؤشر الفارة Mouse موجود في مكان إدخال الأوامر الجديدة حتي يمكنك كتابة الأوامر مباشرة حيث نريد كتابة الكود التالي:

```
BackColor = System.Drawing.Color.Blue
```

6. تعلمنا في الفصل السابق أن الخاصية BackColor هي المسئولة عن تغيير لون خلفية النموذج Form ، وتم تغيير لون الخلفية ليصبح اللون الأزرق والممثل بالقيمة System.Drawing.Color.Blue.
7. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة الضغط بزر الفارة Mouse علي النموذج Form للتأكد من تغيير لون الخلفية إلي اللون الأزرق.

ملحوظة:

حتي لا يتم تكرار الشرح بصورة مبالغ فيها وحتي نقوم بالتركيز علي النقاط الهامة في الشرح ، فسنقوم باتباع الأسلوب الآتي في شرح كيفية معالجة الحدث Event Handling في جميع الأمثلة القادمة:

عندما نذكر أننا نريد إضافة أوامر لأي حدث Event وليكن الحدث MouseUp مثلاً ، فهذا يعني أنك ستقوم بالتأكد من إظهار شاشة الأوامر عن طريق الضغط علي الزر F7 كما تعلمنا في الخطوة الثانية من هذا المثال والتي نقوم فيها باختيار اسم الأداة المطلوب إضافة الحدث Event إليها من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختيار اسم الحدث Event المطلوب وليكن MouseUp من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم نبدأ في كتابة الأوامر المطلوبة كما تعلمنا في الخطوة الخامسة من هذا المثال.

8. نريد الآن تغيير لون خلفية النموذج BackColor عندما يحرر المستخدم زر الفارة Mouse وهذا يعني أننا نريد إنشاء رد فعل للحدث MouseUp ولذلك قم باختيار (Base Class Events) من سهم

اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث MouseUp من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
BackColor = System.Drawing.Color.Yellow
```

9. تم تغيير لون الخلفية ليصبح اللون الأصفر والممثل بالقيمة System.Drawing.Color.Yellow.
10. يمكنك الآن تنفيذ التطبيق عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة الضغط بزر الفارة Mouse علي النموذج Form للتأكد من تغيير لون الخلفية إلي اللون الأزرق ، وعندما تحرر زر الفارة Mouse فإن لون الخلفية يتغير إلي اللون الأصفر.
11. نريد الآن تغيير لون خلفية النموذج BackColor عندما يدخل مؤشر الفارة Mouse داخل مساحة النموذج Form كما نريد تغيير شكل مؤشر الفارة Mouse ، وهذا يعني أننا نريد إنشاء رد فعل للحدث MouseEnter ولذلك قم باختيار (Base Class Events) من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث MouseEnter من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
BackColor = System.Drawing.Color.Red
Cursor = System.Windows.Forms.Cursors.NoMove2D
```

12. تم تغيير لـون الخلفيـة ليصـبح اللـون الأحمـر والممثـل بالقيمـة System.Drawing.Color.Red كما تم تغيير شكل مؤشـر الفـارة Mouse ليصــبح الشــكل الممثــل بالقيمــة System.Windows.Forms.Cursors.NoMove2D.
13. يمكنك الآن تنفيذ التطبيق عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة إدخال مؤشر الفارة Mouse في مساحة النموذج Form للتأكد من تغيير شكل مؤشر الفارة Mouse ومن تغيير لـون خلفيـة النموذج Form.
14. نريد الآن تغيير لون خلفية النموذج BackColor عندما يخرج مؤشـر الفارة Mouse من مساحة النموذج Form ، وهذا يعني أننا نريد إنشاء رد فعل للحدث MouseLeave ولذلك قـم باختيـار (Base Class Events) من سهم اختيارات "اسم الفصيلة "Class Name الموضـح في شكل 1 السابق ثم اختر الحدث MouseLeave من سهم اختيارات "اسم الدالة "Method Name الموضح في شكل 1 السـابق ثـم قـم بإضافة الأوامر التالية.

```
BackColor = System.Drawing.Color.Purple
```

15. تم تغيير لون الخلفيـة ليصـبح اللـون البنفسـجي والممثـل بالقيمـة System.Drawing.Color.Purple.
16. يمكنك الآن تنفيذ التطبيق عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة إدخال مؤشر الفارة Mouse في مساحة النموذج Form

ثم إخراج مؤشر الفارة Mouse من مساحة النموذج Form للتأكد من تغيير لون الخلفية.

17. نريد الآن تغيير لون خلفية النموذج BackColor عندما يتحرك مؤشر الفارة Mouse في مساحة النموذج Form ، وهذا يعني أننا نريد إنشاء رد فعل للحدث MouseMove ولذلك قم باختيار (Base Class Events) من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث MouseMove من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim x As Integer = e.X
Dim y As Integer = e.Y

If (x > 255) Then
        x = 255
End If

If (y > 255) Then
        y = 255
End If

BackColor = System.Drawing.Color.FromArgb(x, y, 0)
```

18. تم معرفة الإحداثي الأفقي للمؤشر عن طريق المتغير e.X (لاحظ أن المعامل Parameter الثاني للدالة Methods التي أنشأتها لغة Visual Basic للحدث MouseMove هو عبارة عن الهدف e الذي من نوع الفصيلة System.Windows.Forms.MouseEventArgs

والمستخدم لمعرفة معلومات عن موضع المؤشر) وبالمثل تم معرفة الإحداثي الرأسي للمؤشر عن طريق المتغير e.Y ثم يتم التحقق من عدم تجاوز أي من القيمتين x أو y للقيمة 255 (والتي تمثل أعلي قيمة مستخدمة في تحديد الألوان) ثم يتم تغيير لون الخلفية ليصبح اللون الممثل بالقيم RGB (أي الأحمر والأخضر والأزرق) حيث سيكون اللون الجديد خليطاً من هذه الألوان الثلاثة.

19. يمكنك الآن تنفيذ التطبيق عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة تحريك مؤشر الفارة Mouse في مساحة النموذج Form للتأكد من تغيير لون الخلفية بشكل مستمر مع حركة المؤشر.
20. نريد الآن إظهار رسالة ترحيب للمستخدم عند فتح النموذج Form ، وهذا يعني أننا نريد إنشاء رد فعل للحدث Load ولذلك قم باختيار (Base Class Events) من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث Load من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
MessageBox.Show("Welcome to my application", "Hello") .
children.Enabled = False
number.Enabled = False
```

21. يتم إظهار الرسالة المطلوبة عن طريق استخدام الدالة Show() وهي أحد دوال Methods الفصيلة MessageBox ، وتستقبل هذه الدالة Method معاملين Parameters: يحدد المعامل الأول Parameter الرسالة التي نريد إظهارها ، ويحدد المعامل الثاني Parameter عنوان

الرسالة Title ، ثم يتم إيقاف عمل أداة صندوق التحقق CheckBox الخاصة بالأولاد وبالمثل لأداة النص Textbox الخاصة بعدد الأولاد لأنه من المنطقي أن يكون الشخص متزوجاً قبل أن يكون له أولاد ولذلك يتم إيقاف عمل هاتين الأداتين للتأكد من أن الشخص متزوج أولاً مع ملاحظة أن أداة صندوق التحقق CheckBox الخاصة بالأولاد يمثلها المتغير children والذي سبق لنا تحديده من خلال الخاصية (Name) كما سبق لنا شرحه في الفصل السابق وبالمثل بالنسبة لأداة النص Textbox الخاصة بعدد الأولاد.

22. يمكنك الآن تنفيذ التطبيق عن طريق الضغط على الزرين Ctrl + F5 ويمكنك التأكد من ظهور الرسالة المطلوبة عند فتح النموذج Form.

## ثانياً: أحداث أداة العنوان Label Events:

تتوافر العديد من الأحداث Events لهذه الأداة ولكن حيث أن أداة العنوان Label هي أداة للقراءة فقط ، فلذلك نجد أن أحداث Events هذه الأداة بلا أهمية عملية ولذلك فلن نقوم بتضييع الوقت في مثال غير عملي وسنقوم بشرح أحداث Events الأداة التالية مباشرة.

## ثالثاً: أحداث أداة النص TextBox Events:

تتوافر الأحداث Events لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الحدث Event | وقت الحدوث |
| --- | --- |
| MouseDown | عندما يضغط المستخدم علي زر الفارة Mouse داخل أداة النص TextBox. |

| | |
|---|---|
| MouseEnter. | عندما يدخل مؤشر الفارة Mouse داخل مساحة أداة النص TextBox. |
| MouseHover | عندما يقف المستخدم بمؤشر الفارة Mouse لمدة ثواني داخل أداة النص TextBox. |
| MouseLeave | عندما يخرج مؤشر الفارة Mouse من مساحة أداة النص TextBox. |
| MouseMove | عندما يحرك المستخدم مؤشر الفارة Mouse داخل مساحة أداة النص TextBox. |
| MouseUp | عندما يحرر المستخدم زر الفارة Mouse بعد الضغط عليه داخل أداة النص TextBox. |
| KeyPress | عندما ينتهي المستخدم من الضغط علي زر معين من لوحة المفاتيح Keyboard. |

سنقوم في المثال التالي بتوضيح كيفية استخدام أحداث Events هذه الأداة.

**خطوات التنفيذ:**

1. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط علي الزر F7).
2. نريد الآن نقل مؤشر الفارة Mouse لأداة النص TextBox الخاصة بالعنوان عندما يضغط المستخدم علي زر الإدخال Enter (وهذه العملية معروفة باسم Focusing) وهذا يعني أننا نريد إنشاء رد فعل للحدث KeyPress ولذلك قم باختيار myName من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث

KeyPress من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim c As Char = e.KeyChar

If (c = vbNewLine.Chars(0)) Then
    myAddress.Focus()
End If
```

3. تم معرفة الحرف الذي ضغطه المستخدم عن طريق المتغير e.KeyChar (لاحظ أن المعامل Parameter الثاني للدالة Methods التي أنشأتها لغة Visual Basic للحدث KeyPress هو عبارة عن الهدف e الذي من نوع الفصيلة System.Windows.Forms.KeyPressEventArgs والمستخدم لمعرفة معلومات عن الحرف الذي تم ضغطه من لوحة المفاتيح Keyboard) ثم يتم مقارنة الحرف المضغوط بزر الإدخال Enter (والممثل بالحرف الأول من القيمة vbNewLine والتي تمثل سطر جديد) ، فإذا كان الحرف المضغوط هو زر الإدخال Enter فعلاً ، فإنه يتم نقل مؤشر الفارة Mouse لأداة النص TextBox الخاصة بالعنوان والممثلة بالمتغير myAddress (راجع الفصل السابق) عن طريق الدالة Focus().
4. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة الكتابة في أداة النص TextBox الخاصة بالاسم ثم الضغط علي زر الإدخال Enter للتأكد من نقل مؤشر الفارة Mouse لأداة النص TextBox الخاصة بالعنوان.

## رابعاً: أحداث أداة قائمة الاختيارات ComboBox:

تتوافر الأحداث Events لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الحدث Event | وقت الحدوث |
|---|---|
| MouseDown | عندما يضغط المستخدم علي زر الفارة Mouse داخل مساحة أداة قائمة الاختيارات ComboBox. |
| MouseEnter | عندما يدخل مؤشر الفارة Mouse داخل مساحة أداة قائمة الاختيارات ComboBox. |
| MouseHover | عندما يقف المستخدم بمؤشر الفارة Mouse لمدة ثواني داخل مساحة أداة قائمة الاختيارات ComboBox. |
| MouseLeave | عندما يخرج مؤشر الفارة Mouse من مساحة أداة قائمة الاختيارات ComboBox. |
| MouseMove | عندما يحرك المستخدم مؤشر الفارة Mouse داخل مساحة أداة قائمة الاختيارات ComboBox. |
| MouseUp | عندما يحرر المستخدم زر الفارة Mouse بعد الضغط عليه داخل مساحة أداة قائمة الاختيارات ComboBox. |
| SelectedIndexChanged | عندما يقوم المستخدم بتغيير الاختيار من قائمة الاختيارات المتاحة. |

سنقوم في المثال التالي بتوضيح كيفية استخدام أحداث Events هذه الأداة.

**خطوات التنفيذ:**

1. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط علي الزر F7).
2. نريد الآن إظهار رسالة للمستخدم لتبين اختياره عندما يقوم بتغيير الدولة من أداة قائمة الاختيارات ComboBox وهذا يعني أننا نريد إنشاء رد فعل للحدث SelectedIndexChanged ولذلك قم باختيار myCountry من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث SelectedIndexChanged من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim s As String = myCountry.Text

MessageBox.Show("You are from " & s , " Country")
```

3. تم معرفة اختيار المستخدم عن طريق المتغير myCountry.Text (لاحظ أن أداة قائمة الاختيارات ComboBox الخاصة بالدولة ممثلة بالمتغير myCountry كما شرحنا في الفصل السابق) ثم يتم إظهار الرسالة المطلوبة عن طريق استخدام الدالة Show() وهي أحد دوال Methods الفصيلة MessageBox ، وتستقبل هذه الدالة Method معاملين Parameters: يحدد المعامل الأول Parameter الرسالة التي نريد إظهارها ، ويحدد المعامل الثاني Parameter عنوان الرسالة Title.
4. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين

Ctrl + F5 ويمكنك تجربة تغيير الدولة فـــي أداة قائمـــة الاختيـــارات ComboBox الخاصة بالدولة وتأكد من ظهور رسالة لتبين اسم الدولة المختارة.

## خامساً: أحداث أداة زر الراديو RadioButton Events:

تتوافر الأحداث Events لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الحدث Event | وقت الحدوث |
|---|---|
| MouseDown | عندما يضغط المستخدم علي زر الفارة Mouse داخل مساحة أداة زر الراديو RadioButton. |
| MouseEnter | عندما يدخل مؤشر الفارة Mouse داخل مساحة أداة زر الراديو RadioButton. |
| MouseHover | عندما يقف المستخدم بمؤشر الفارة Mouse لمدة ثواني داخل مساحة أداة زر الراديو RadioButton. |
| MouseLeave | عندما يخرج مؤشر الفارة Mouse من مساحة أداة زر الراديو RadioButton. |
| MouseMove | عندما يحرك المستخدم مؤشر الفارة Mouse داخل مساحة أداة زر الراديو RadioButton. |
| MouseUp | عندما يحرر المستخدم زر الفارة Mouse بعد الضغط عليه داخل مساحة أداة زر الراديو RadioButton. |
| CheckedChanged | عندما يقوم المستخدم بتغيير اختيار أداة زر الراديو RadioButton. |

سنقوم في المثال التالي بتوضيح كيفية استخدام أحداث Events هذه الأداة.

**خطوات التنفيذ:**

1. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط علي الزر F7).
2. نريد الآن إيقاف عمل أداة صندوق التحقق CheckBox الخاصة بالأولاد وبالمثل لأداة النص Textbox الخاصة بعدد الأولاد عندما يقوم المستخدم بالضغط علي أداة زر الراديو RadioButton الخاصة بالشخص الأعزب لاختيارها وهذا يعني أننا نريد إنشاء رد فعل للحدث CheckedChanged ولذلك قم باختيار single1 من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث CheckedChanged من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim b As Boolean = single1.Checked

If (b = True) Then
        children.Enabled = False
        number.Enabled = False
End If
```

3. تم معرفة ما إذا كان المستخدم قد ضغط علي أداة زر الراديو RadioButton الخاصة بالشخص الأعزب عن طريق المتغير single.Checked (لاحظ أن أداة زر الراديو RadioButton

الخاصة بالشخص الأعزب ممثلة بالمتغير single كما شرحنا في الفصل السابق) ثم يتم التأكد من أن المستخدم قد ضغط علي أداة زر الراديو RadioButton الخاصة بالشخص الأعزب (وذلك إذا كانت قيمة المتغير b تساوي القيمة True) ، فإذا تحقق هذا الشرط فإنه يتم إيقاف عمل أداة صندوق التحقق CheckBox الخاصة بالأولاد وبالمثل لأداة النص Textbox الخاصة بعدد الأولاد عن طريق تغيير الخاصية Enabled لتكون مساوية للقيمة False.

4. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط علي الزر F7).

5. نريد الآن إعادة تمكين Enable عمل أداة صندوق التحقق CheckBox الخاصة بالأولاد عندما يقوم المستخدم بالضغط علي أداة زر الراديو RadioButton الخاصة بالشخص المتزوج لاختيارها وهذا يعني أننا نريد إنشاء رد فعل للحدث CheckedChanged ولذلك قم باختيار married من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث CheckedChanged من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim b As Boolean = married.Checked

If (b = True) Then
        children.Enabled = True
End If
```

6. تم معرفة ما إذا كان المستخدم قد ضغط علي أداة زر الراديو RadioButton الخاصة بالشخص المتزوج عن طريق المتغير married.Checked (لاحظ أن أداة زر الراديو RadioButton الخاصة بالشخص المتزوج ممثلة بالمتغير married كما شرحنا في الفصل السابق) ثم يتم التأكد من أن المستخدم قد ضغط علي أداة زر الراديو RadioButton الخاصة بالشخص المتزوج (وذلك إذا كانت قيمة المتغير b تساوي القيمة True) ، فإذا تحقق هذا الشرط فإنه يتم إعادة تمكين Enable عمل أداة صندوق التحقق CheckBox الخاصة بالأولاد عن طريق تغيير الخاصية Enabled لتكون مساوية للقيمة True.
7. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة تغيير الحالة الاجتماعية للشخص لتكون متزوجاً مرة وأعزب مرة أخري وتأكد من إيقاف عمل الأدوات المطلوبة وإعادة تمكينها Enable علي حسب اختيار المستخدم.

## سادساً: أحداث أداة صندوق التحقق CheckBox Events:

تتوافر الأحداث Events لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الحدث Event | وقت الحدوث |
|---|---|
| MouseDown | عندما يضغط المستخدم علي زر الفارة Mouse داخل مساحة أداة صندوق التحقق CheckBox. |
| MouseEnter | عندما يدخل مؤشر الفارة Mouse داخل مساحة أداة صندوق التحقق CheckBox. |

| | |
|---|---|
| MouseHover | عندما يقف المستخدم بمؤشر الفارة Mouse لمدة ثواني داخل مساحة أداة صندوق التحقق CheckBox. |
| MouseLeave | عندما يخرج مؤشر الفارة Mouse من مساحة أداة صندوق التحقق CheckBox. |
| MouseMove | عندما يحرك المستخدم مؤشر الفارة Mouse داخل مساحة أداة صندوق التحقق CheckBox. |
| MouseUp | عندما يحرر المستخدم زر الفارة Mouse بعد الضغط عليه داخل مساحة أداة صندوق التحقق CheckBox. |
| CheckedChanged | عندما يقوم المستخدم بتغيير اختيار أداة صندوق التحقق CheckBox. |

سنقوم في المثال التالي بتوضيح كيفية استخدام أحداث Events هذه الأداة.

**خطوات التنفيذ:**

1. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط علي الزر F7).
2. نريد الآن تمكين Enable عمل أداة النص Textbox الخاصة بعدد الأولاد عندما يقوم المستخدم بالضغط علي أداة صندوق التحقق CheckBox الخاصة بالأولاد لاختيارها وهذا يعني أننا نريد إنشاء رد فعل للحدث CheckedChanged ولذلك قم باختيار children من

سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث CheckedChanged من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim b1 As Boolean = children.Checked
Dim b2 As Boolean = married.Checked

If (b1 = True And b2 = True) Then
        number.Enabled = True
End If
```

3. تم معرفة ما إذا كان المستخدم قد ضغط علي أداة صندوق التحقق CheckBox الخاصة بالأولاد عن طريق المتغير children.Checked (لاحظ أن أداة صندوق التحقق CheckBox الخاصة بالأولاد ممثلة بالمتغير children كما شرحنا في الفصل السابق) كما يتم معرفة ما إذا كان المستخدم قد ضغط علي أداة زر الراديو RadioButton الخاصة بالشخص المتزوج عن طريق المتغير married.Checked ، (ويتحقق هذان الشرطان إذا كانت قيمتي المتغيرين b1 و b2 تساويان القيمة True) ، فإذا تحقق هذان الشرطان فإنه يتم تمكين Enable عمل أداة النص Textbox الخاصة بعدد الأولاد عن طريق تغيير الخاصية Enabled لتكون مساوية للقيمة True.

## سابعاً: أداة الزر Button:

تتوافر الأحداث Events لهذه الأداة كما هو واضح في الجدول التالي.

| اسم الحدث Event | وقت الحدوث |
|---|---|
| MouseDown | عندما يضغط المستخدم علي زر الفارة Mouse داخل مساحة أداة الزر Button. |
| MouseEnter | عندما يدخل مؤشر الفارة Mouse داخل مساحة أداة الزر Button. |
| MouseHover | عندما يقف المستخدم بمؤشر الفارة Mouse لمدة ثواني داخل مساحة أداة الزر Button. |
| MouseLeave | عندما يخرج مؤشر الفارة Mouse من مساحة أداة الزر Button. |
| MouseMove | عندما يحرك المستخدم مؤشر الفارة Mouse داخل مساحة أداة الزر Button. |
| MouseUp | عندما يحرر المستخدم زر الفارة Mouse بعد الضغط عليه داخل مساحة أداة الزر Button. |
| Click | عندما يقوم المستخدم بالضغط علي أداة الزر Button. |

سنقوم في المثال التالي بتوضيح كيفية استخدام أحداث Events هذه الأداة.

## خطوات التنفيذ:

1. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط على الزر F7).
2. نريد الآن إغلاق نافذة البرنامج عندما يقوم المستخدم بالضغط على أداة الزر Button المسمى Exit وهذا يعني أننا نريد إنشاء رد فعل للحدث Click ولذلك قم باختيار quit من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث Click من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Application.Exit()
```

3. تم إغلاق نافذة البرنامج عن طريق استخدام الدالة Exit() وهي أحد الدوال Methods في الفصيلة Application.
4. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة الضغط على أداة الزر Button المسمي Exit وتأكد من إغلاق نافذة البرنامج.
5. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط على الزر F7).
6. نريد الآن مسح جميع اختيارات المستخدم عندما يقوم المستخدم بالضغط على أداة الزر Button المسمي Clear وهذا يعني أننا نريد إنشاء رد فعل للحدث Click ولذلك قم باختيار clear من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث

Click من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
myName.Text = ""
myAddress.Text = ""
myCountry.Text = ""
single1.Checked = False
married.Checked = False
children.Checked = False
number.Text = ""
myName.Focus()
```

7. تم مسح جميع البيانات عن طريق استخدام الخاصية المناسبة لكل أداة ثم يتم وضع مؤشر الفارة Mouse في أداة النص TextBox الخاصة بالاسم.
8. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة كتابة بيانات في النموذج Form ثم الضغط علي أداة الزر Button المسمي Clear وتأكد من مسح جميع البيانات.
9. تأكد من إظهار شاشة الأوامر (والتي أظهرناها عن طريق الضغط علي الزر F7).
10. نريد الآن إظهار رسالة لتبين جميع اختيارات المستخدم عندما يقوم المستخدم بالضغط علي أداة الزر Button المسمي Report وهذا يعني أننا نريد إنشاء رد فعل للحدث Click ولذلك قم باختيار report من سهم اختيارات "اسم الفصيلة Class Name" الموضح في شكل 1 السابق ثم اختر الحدث Click من سهم اختيارات "اسم الدالة Method Name" الموضح في شكل 1 السابق ثم قم بإضافة الأوامر التالية.

```
Dim s As String = "Name is " & myName.Text & vbCrLf
s &= "Address is " & myAddress.Text & vbCrLf
s &= "Country is " & myCountry.Text & vbCrLf
s &= "Single is: " & single1.Checked & vbCrLf
s &= "Married is: " & married.Checked & vbCrLf
s &= "Have Children: " & children.Checked & vbCrLf
s &= "Number of children is " & number.Text & vbCrLf

MessageBox.Show(s, "Report")
```

11. تم معرفة جميع البيانات عن طريق استخدام الخاصية المناسبة لكل أداة ثم يتم إظهار رسالة لتبين جميع اختيارات المستخدم مع ملاحظة استخدام القيمة vbCrLf والتي تبين وجود سطر جديد.
12. نحتاج الآن لتنفيذ التطبيق ، ويتم ذلك عن طريق الضغط على الزرين Ctrl + F5 ويمكنك تجربة كتابة بيانات في النموذج Form ثم الضغط علي أداة الزر Button المسمي Report وتأكد من إظهار رسالة تبين جميع اختيارات المستخدم.

**الكود الكامل:**

فيما يلي تجد الأوامر الكاملة للملف بعد إنشاء أوامر معالجة الأحداث Events Handling لجميع الأدوات السابق شرحها حتي يمكنك التأكد من صحة أوامرك مع ملاحظة أن أوامر معالجة الأحداث Events Handling التي أنشأناها في هذا الفصل موجودة في السطور من 225 إلي 337.

```
1:  Public Class myData
2:      Inherits System.Windows.Forms.Form
3:
4:  #Region " Windows Form Designer generated code "
5:
```

```
6:    Public Sub New()
7:       MyBase.New()
8:
9:       'This call is required by the Windows Form
   Designer.
10:      InitializeComponent()
11:
12:      'Add any initialization after the
   InitializeComponent() call
13:
14:   End Sub
15:
16:   'Form overrides dispose to clean up the component
   list.
17:   Protected Overloads Overrides Sub Dispose(ByVal
   disposing As Boolean)
18:      If disposing Then
19:         If Not (components Is Nothing) Then
20:            components.Dispose()
21:         End If
22:      End If
23:      MyBase.Dispose(disposing)
24:   End Sub
25:
26:   'Required by the Windows Form Designer
27:   Private components As
   System.ComponentModel.IContainer
28:
29:   'NOTE: The following procedure is required by the
   Windows Form Designer
30:   'It can be modified using the Windows Form
   Designer.
```

```
31:    'Do not modify it using the code editor.
32:    Friend WithEvents Label1 As System.Windows.Forms.Label
33:    Friend WithEvents myName As System.Windows.Forms.TextBox
34:    Friend WithEvents myAddress As System.Windows.Forms.TextBox
35:    Friend WithEvents Label2 As System.Windows.Forms.Label
36:    Friend WithEvents Label3 As System.Windows.Forms.Label
37:    Friend WithEvents myCountry As System.Windows.Forms.ComboBox
38:    Friend WithEvents Label4 As System.Windows.Forms.Label
39:    Friend WithEvents single1 As System.Windows.Forms.RadioButton
40:    Friend WithEvents married As System.Windows.Forms.RadioButton
41:    Friend WithEvents children As System.Windows.Forms.CheckBox
42:    Friend WithEvents Label5 As System.Windows.Forms.Label
43:    Friend WithEvents number As System.Windows.Forms.TextBox
44:    Friend WithEvents report As System.Windows.Forms.Button
45:    Friend WithEvents clear As System.Windows.Forms.Button
46:    Friend WithEvents quit As System.Windows.Forms.Button
```

```
47:   <System.Diagnostics.DebuggerStepThrough()>
   Private Sub InitializeComponent()
48:       Me.Label1 = New System.Windows.Forms.Label()
49:       Me.myName                =                New
   System.Windows.Forms.TextBox()
50:       Me.myAddress             =                New
   System.Windows.Forms.TextBox()
51:       Me.Label2 = New System.Windows.Forms.Label()
52:       Me.Label3 = New System.Windows.Forms.Label()
53:       Me.myCountry             =                New
   System.Windows.Forms.ComboBox()
54:       Me.Label4 = New System.Windows.Forms.Label()
55:       Me.single1               =                New
   System.Windows.Forms.RadioButton()
56:       Me.married               =                New
   System.Windows.Forms.RadioButton()
57:       Me.children              =                New
   System.Windows.Forms.CheckBox()
58:       Me.Label5 = New System.Windows.Forms.Label()
59:       Me.number                =                New
   System.Windows.Forms.TextBox()
60:       Me.report                =                New
   System.Windows.Forms.Button()
61:       Me:clear = New System.Windows.Forms.Button()
62:       Me.quit = New System.Windows.Forms.Button()
63:       Me.SuspendLayout()
64:       '
65:       'Label1
66:       '
67:       Me.Label1.AutoSize = True
68:       Me.Label1.Font           =                New
   System.Drawing.Font("Palatino Linotype", 9.75!)
```

```
69:     Me.Label1.ForeColor                          =
    System.Drawing.Color.Blue
70:     Me.Label1.Location          =           New
    System.Drawing.Point(56, 40)
71:     Me.Label1.Name = "Label1"
72:     Me.Label1.Size = New System.Drawing.Size(44,
    18)
73:     Me.Label1.TabIndex = 0
74:     Me.Label1.Text = "Name:"
75:     '
76:     'myName
77:     '
78:     Me.myName.BackColor                          =
    System.Drawing.Color.FromArgb(CType(255,    Byte),
    CType(255, Byte), CType(192, Byte))
79:     Me.myName.Font              =           New
    System.Drawing.Font("Palatino    Linotype",    9.75!,
    System.Drawing.FontStyle.Regular,
    System.Drawing.GraphicsUnit.Point, CType(0, Byte))
80:     Me.myName.ForeColor                          =
    System.Drawing.Color.Blue
81:     Me.myName.Location          =           New
    System.Drawing.Point(170, 40)
82:     Me.myName.MaxLength = 15
83:     Me.myName.Name = "myName"
84:     Me.myName.TabIndex = 1
85:     Me.myName.Text = "Enter Name"
86:     Me.myName.TextAlign                          =
    System.Windows.Forms.HorizontalAlignment.Center
87:     '
88:     'myAddress
89:     '
```

```
90:       Me.myAddress.BackColor                              =
   System.Drawing.Color.FromArgb(CType(255,     Byte),
   CType(255, Byte), CType(192, Byte))
91:       Me.myAddress.ForeColor                              =
   System.Drawing.Color.Blue
92:       Me.myAddress.HideSelection = False
93:       Me.myAddress.Location            =              New
   System.Drawing.Point(170, 100)
94:       Me.myAddress.Multiline = True
95:       Me.myAddress.Name = "myAddress"
96:       Me.myAddress.ScrollBars                             =
   System.Windows.Forms.ScrollBars.Both
97:       Me.myAddress.Size             =                 New
   System.Drawing.Size(200, 100)
98:       Me.myAddress.TabIndex = 2
99:       Me.myAddress.Text = ""
100:          '
101:          'Label2
102:          '
103:          Me.Label2.AutoSize = True
104:          Me.Label2.Font             =               New
   System.Drawing.Font("Palatino Linotype", 9.75!)
105:          Me.Label2.ForeColor                             =
   System.Drawing.Color.Blue
106:          Me.Label2.Location             =             New
   System.Drawing.Point(56, 100)
107:          Me.Label2.Name = "Label2"
108:          Me.Label2.Size             =                 New
   System.Drawing.Size(58, 18)
109:          Me.Label2.TabIndex = 3
110:          Me.Label2.Text = "Address:"
111:          '
```

```
112:        'Label3
113:        '
114:        Me.Label3.AutoSize = True
115:        Me.Label3.Font          =          New
   System.Drawing.Font("Palatino Linotype", 9.75!)
116:        Me.Label3.ForeColor                 =
   System.Drawing.Color.Blue
117:        Me.Label3.Location          =       New
   System.Drawing.Point(56, 230)
118:        Me.Label3.Name = "Label3"
119:        Me.Label3.Size          =          New
   System.Drawing.Size(55, 18)
120:        Me.Label3.TabIndex = 4
121:        Me.Label3.Text = "Country"
122:        '
123:        'myCountry
124:        '
125:        Me.myCountry.Items.AddRange(New  Object()
   {"Egypt", "Saudi Arabia", "Lebanon", "Morocco"})
126:        Me.myCountry.Location          =    New
   System.Drawing.Point(170, 230)
127:        Me.myCountry.Name = "myCountry"
128:        Me.myCountry.Size          =        New
   System.Drawing.Size(150, 21)
129:        Me.myCountry.TabIndex = 5
130:        '
131:        'Label4
132:        '
133:        Me.Label4.AutoSize = True
134:        Me.Label4.Font          =          New
   System.Drawing.Font("Palatino Linotype", 9.75!)
```

```
135:        Me.Label4.ForeColor                                    =
   System.Drawing.Color.Blue
136:        Me.Label4.Location              =               New
   System.Drawing.Point(56, 290)
137:        Me.Label4.Name = "Label4"
138:        Me.Label4.Size              =                   New
   System.Drawing.Size(91, 18)
139:        Me.Label4.TabIndex = 6
140:        Me.Label4.Text = "Marital Status:"
141:        '
142:        'single1
143:        '
144:        Me.single1.Location             =               New
   System.Drawing.Point(170, 290)
145:        Me.single1.Name = "single1"
146:        Me.single1.Size             =                   New
   System.Drawing.Size(60, 24)
147:        Me.single1.TabIndex = 7
148:        Me.single1.Text = "Single"
149:        '
150:        'married
151:        '
152:        Me.married.Location             =               New
   System.Drawing.Point(280, 290)
153:        Me.married.Name = "married"
154:        Me.married.Size             =                   New
   System.Drawing.Size(65, 24)
155:        Me.married.TabIndex = 8
156:        Me.married.Text = "Married"
157:        '
158:        'children
159:        '
```

```
160:        Me.children.Location         =        New
   System.Drawing.Point(390, 290)
161:        Me.children.Name = "children"
162:        Me.children.Size          =        New
   System.Drawing.Size(100, 24)
163:        Me.children.TabIndex = 9
164:        Me.children.Text = "Have Children"
165:        '
166:        'Label5
167:        '
168:        Me.Label5.AutoSize = True
169:        Me.Label5.Font          =        New
   System.Drawing.Font("Palatino Linotype", 9.75!)
170:        Me.Label5.ForeColor                      =
   System.Drawing.Color.Blue
171:        Me.Label5.Location          =        New
   System.Drawing.Point(390, 330)
172:        Me.Label5.Name = "Label5"
173:        Me.Label5.Size          =        New
   System.Drawing.Size(130, 18)
174:        Me.Label5.TabIndex = 10
175:        Me.Label5.Text = "Number of Children:"
176:        '
177:        'number
178:        '
179:        Me.number.Location          =        New
   System.Drawing.Point(552, 330)
180:        Me.number.Name = "number"
181:        Me.number.Size          =        New
   System.Drawing.Size(30, 20)
182:        Me.number.TabIndex = 11
183:        Me.number.Text = ""
```

```
184:        '
185:        'report
186:        '
187:        Me.report.Location          =          New
   System.Drawing.Point(56, 390)
188:        Me.report.Name = "report"
189:        Me.report.Size          =          New
   System.Drawing.Size(75, 25)
190:        Me.report.TabIndex = 12
191:        Me.report.Text = "Report"
192:        '
193:        'clear
194:        '
195:        Me.clear.Location          =          New
   System.Drawing.Point(204, 390)
196:        Me.clear.Name = "clear"
197:        Me.clear.Size = New System.Drawing.Size(75,
   25)
198:        Me.clear.TabIndex = 13
199:        Me.clear.Text = "Clear"
200:        '
201:        'quit
202:        '
203:        Me.quit.Location          =          New
   System.Drawing.Point(352, 390)
204:        Me.quit.Name = "quit"
205:        Me.quit.Size = New System.Drawing.Size(75,
   25)
206:        Me.quit.TabIndex = 14
207:        Me.quit.Text = "Exit"
208:        '
209:        'myData
```

```
210:        '
211:        Me.AutoScaleBaseSize        =        New
    System.Drawing.Size(5, 13)
212:        Me.BackColor                              =
    System.Drawing.Color.FromArgb(CType(224,    Byte),
    CType(224, Byte), CType(224, Byte))
213:        Me.ClientSize          =          New
    System.Drawing.Size(792, 573)
214:        Me.Controls.AddRange(New
    System.Windows.Forms.Control() {Me.quit, Me.clear,
    Me.report,   Me.number,   Me.Label5,   Me.children,
    Me.married,  Me.single1,  Me.Label4,  Me.myCountry,
    Me.Label3, Me.Label2, Me.myAddress, Me.myName,
    Me.Label1})
215:        Me.Cursor                                 =
    System.Windows.Forms.Cursors.Hand
216:        Me.Name = "myData"
217:        Me.StartPosition                          =
    System.Windows.Forms.FormStartPosition.CenterScree
    n
218:        Me.Text = "My Data"
219:        Me.ResumeLayout(False)
220:
221:     End Sub
222:
223:  #End Region
224:
225:     Private Sub myData_MouseDown(ByVal sender
    As        Object,        ByVal        e        As
    System.Windows.Forms.MouseEventArgs)     Handles
    MyBase.MouseDown
226:        BackColor = System.Drawing.Color.Blue
```

```
227:        End Sub
228:
229:        Private Sub myData_MouseUp(ByVal sender As
    Object,             ByVal             e             As
    System.Windows.Forms.MouseEventArgs)      Handles
    MyBase.MouseUp
230:            BackColor = System.Drawing.Color.Yellow
231:        End Sub
232:
233:        Private Sub myData_MouseEnter(ByVal sender
    As Object, ByVal e As System.EventArgs) Handles
    MyBase.MouseEnter
234:            BackColor = System.Drawing.Color.Red
235:            Cursor                                  =
    System.Windows.Forms.Cursors.NoMove2D
236:        End Sub
237:
238:        Private Sub myData_MouseLeave(ByVal sender
    As Object, ByVal e As System.EventArgs) Handles
    MyBase.MouseLeave
239:            BackColor = System.Drawing.Color.Purple
240:        End Sub
241:
242:        Private Sub myData_MouseMove(ByVal sender
    As          Object,          ByVal          e          As
    System.Windows.Forms.MouseEventArgs)      Handles
    MyBase.MouseMove
243:            Dim x As Integer = e.X
244:            Dim y As Integer = e.Y
245:
246:            If (x > 255) Then
247:                x = 255
```

```
248:         End If
249:
250:         If (y > 255) Then
251:            y = 255
252:         End If
253:
254:         BackColor                                =
   System.Drawing.Color.FromArgb(x, y, 0)
255:
256:      End Sub
257:
258:      Private Sub myData_Load(ByVal sender As
   Object, ByVal e As System.EventArgs) Handles
   MyBase.Load
259:         MessageBox.Show("Welcome       to      my
   application", "Hello")
260:         children.Enabled = False
261:         number.Enabled = False
262:
263:      End Sub
264:
265:      Private Sub myName_KeyPress(ByVal sender As
   Object,            ByVal            e            As
   System.Windows.Forms.KeyPressEventArgs) Handles
   myName.KeyPress
266:         Dim c As Char = e.KeyChar
267:
268:         If (c = vbNewLine.Chars(0)) Then
269:            myAddress.Focus()
270:         End If
271:
272:      End Sub
```

```
273:
274:      Private                                          Sub
   myCountry_SelectedIndexChanged(ByVal  sender  As
   Object,  ByVal  e  As  System.EventArgs)  Handles
   myCountry.SelectedIndexChanged
275:          Dim s As String = myCountry.Text
276:
277:          MessageBox.Show("You are from " + s + "
   Country")
278:
279:      End Sub
280:
281:      Private  Sub  single1_CheckedChanged(ByVal
   sender  As  Object,  ByVal  e  As  System.EventArgs)
   Handles single1.CheckedChanged
282:          Dim b As Boolean = single1.Checked
283:
284:          If (b = True) Then
285:             children.Enabled = False
286:             number.Enabled = False
287:          End If
288:
289:      End Sub
290:
291:      Private  Sub  married_CheckedChanged(ByVal
   sender  As  Object,  ByVal  e  As  System.EventArgs)
   Handles married.CheckedChanged
292:          Dim b As Boolean = married.Checked
293:
294:          If (b = True) Then
295:             children.Enabled = True
296:          End If
```

```
297:
298:      End Sub
299:
300:      Private Sub children_CheckedChanged(ByVal
    sender As Object, ByVal e As System.EventArgs)
    Handles children.CheckedChanged
301:          Dim b1 As Boolean = children.Checked
302:          Dim b2 As Boolean = married.Checked
303:
304:          If (b1 = True And b2 = True) Then
305:              number.Enabled = True
306:          End If
307:
308:      End Sub
309:
310:      Private Sub quit_Click(ByVal sender As Object,
    ByVal e As System.EventArgs) Handles quit.Click
311:          Application.Exit()
312:      End Sub
313:
314:      Private Sub clear_Click(ByVal sender As Object,
    ByVal e As System.EventArgs) Handles clear.Click
315:          myName.Text = ""
316:          myAddress.Text = ""
317:          myCountry.Text = ""
318:          single1.Checked = False
319:          married.Checked = False
320:          children.Checked = False
321:          number.Text = ""
322:          myName.Focus()
323:
324:      End Sub
```

```
325:
326:    Private Sub report_Click(ByVal sender As
  Object, ByVal e As System.EventArgs) Handles
  report.Click
327:        Dim s As String = "Name is " & myName.Text
  & vbCrLf
328:        s &= "Address is " & myAddress.Text &
  vbCrLf
329:        s &= "Country is " & myCountry.Text &
  vbCrLf
330:        s &= "Single is: " & single1.Checked &
  vbCrLf
331:        s &= "Married is: " & married.Checked &
  vbCrLf
332:        s &= "Have Children: " & children.Checked &
  vbCrLf
333:        s &= "Number of children is " & number.Text
  & vbCrLf
334:
335:        MessageBox.Show(s, "Report")
336:
337:    End Sub
338: End Class
```

# الفصل الحادي عشر

## هدف البيانات

## ActiveX Data Object (ADO .NET)

فى هذا الفصل سوف نتعرف علي كيفية التعامـــل مـــع قواعد البيانات Databases في لغـــة VB.net مـــن خلال تقنية NET. ADO وذلك مـــن خـــلال النقـــاط التالية:

1. مقدمة Introduction.
2. ماهى قواعد البيانات الارتباطية Relational Databases؟
3. لغة الإستعلامات المركبة SQL.
4. استخدام فصائل ADO .NET.
5. تطبيق عملى باستخدام ADO .NET.

هدف البيانات
ActiveX Data Object (ADO .NET)

## مقدمة:

فى الفصول السابقة تعلمنا أساسيات لغة VB.net وحان الوقت للتعامل مع واحد من أهم موضوعات البرمجة ألا وهو معالجة قواعد البيانات Databases ، وذلك لأن كثيراً من التطبيقات قائمة على حفظ ومعالجة واسترجاع البيانات الموجودة فى قواعد البيانات Databases ولذلك يجب علينا بداية معرفة ما هى قواعد البيانات Databases وأنواعها.

## قواعد البيانات Databases:

أبسط تعريف لقاعدة البيانات هو أنها "**مجموعة من البيانات المتكاملة**".

ويتم تخزين البيانات فى قاعدة البيانات بشكل منظم وذلك لتسهيل عملية الوصول ومعالجة البيانات الموجودة Data access and manipulation. ويوجد العديد من الطرق لتنظيم البيانات Data Organization داخل قاعدة البيانات وأهم هذه الطرق ما يسمى بـ "نظام إدارة قواعد البيانات DBMS" وهو اختصار لـ "Database Management System". والـ DBMS هو الطريقة "النظام" الذى نستطيع به تخزين وتنظيم البيانات فى شكل مترابط ومتكامل. وتقوم الشركات المختلفة بإنتاج الـ DBMS (يسمى أحيانا "محرك قواعد البيانات) فى شكل برامج مثل أكسس Access وأوراكل Oracle كما سنوضح لاحقاً.

والـ DBMS يقدم للمبرمج خدمة كبيرة حيث أنه يجعل المبرمج قادراً على الوصول access وتخزين البيانات فى قاعدة البيانات Database دونما الاهتمام بكيفية تشكيل وتمثيل البيانات داخل قاعدة البيانات Database.

ويوجد أنواع كثيرة من الـ DBMS تعتمد على نوع قاعدة البيانات Database.

## أنواع قواعد البيانات:

يوجد العديد من أنواع قواعد البيانات Databases ولكن أهمهم هم:

- قواعد البيانات الارتباطية Relational Database .

وهى النوع السائد استعماله حالياً ، ومن أشهر برامج هذا النوع:

1- مايكروسوفت أكسيس Microsoft Access.

2- مايكروسوفت SQL سيرفر Microsoft SQL Server (وهو الذى سنستخدمه فى الأمثلة).

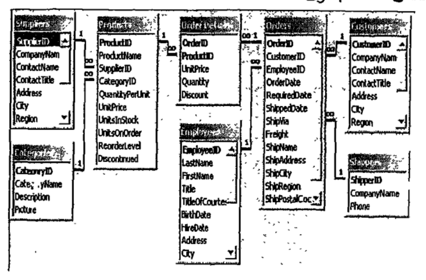
3- أوراكل Oracle.

4- سايبس Sybase.

- قواعد البيانات النصية Textual Database.
- قواعد البيانات الهرمية Hierarchal Database.
- قواعد البيانات المسطحة Flat Database ويعتبر برنامج مايكروسوفت إكسيل أهم أنواع البيانات المسطحة Flat Database.

## قواعد البيانات الارتباطية:

وفى هذا النوع تتكون قاعدة البيانات من جدول table أو أكثر بحيث تكون هذه الجداول مرتبطة ببعضها البعض وذلك كما يتضح من الشكل 1 ، ويتم تخزين البيانات داخل هذه الجداول.




شكل 1 قواعد البيانات الارتباطية

ولقواعد البيانات لغة خاصة بها وهى لغة الاستعلامات المركبة SQL وهى اختصار لـ Structured Query Language.

## لغة SQL:

لغة SQL عبارة عن مجموعة من الأوامر تستخدم فى التعامل مع قواعد البيانات. وتنقسم أوامر لغة SQL إلى فئتين رئيسيتين هى:

**DDL**(Data Definition Language).
**DML**(Data Manipulation Language).

حيث:

- DDL هى مجموعة الأوامر المسئولة عن إنشاء الكيانات المختلفة المكونة لقاعدة البيانات مثل الجداول والإجراءات Stored procedures و... وأيضا قاعدة البيانات نفسها.
- DML هى مجموعة الأوامر المسئولة عن التعامل مع البيانات مثل الاستعلامات والإضافة والحذف والتعديل وهى أكثر الأوامر إستعمالاً ولذلك سنتناولها بشكل سريع.

وسنتعرف فى الفقرة التالية على بعض أوامر لغة DML.

## أوامر لغة DML:

تحتوى لغة DML على أوامر لعرض Displaying واسترجاع Retrieving البيانات وأوامر للتأثير الفعلى فى البيانات مثل الحذف والإضافة والتعديل والجدول التالى يوضح بعض هذه الأوامر.

| الأمر | الوصف |
|---|---|
| Select | هذا الأمر يقوم باسترجاع حقل field واحد أو أكثر من جدول واحد أو أكثر. |
| From | يستخدم هذا المفتاح فى تحديد الجدول الذى نريد التعامل مع |

| | |
|---|---|
| | بياناته وعرضها أو التأثير الفعلى فيها. |
| Where | يستخدم هذا المفتاح فى تحديد شرط ما لرؤية أو التأثير فى سجل ما أو سجلات محددة بناء على الشرط. |
| INSERT | يقوم هذا الأمر بإدخال سجلات جديدة إلى جدول محدد. |
| UPDATE | يقوم هذا الأمر بتحديث البيانات فى جدول محدد. |
| DELETE | يقوم هذا الأمر بحذف بيانات من جدول محدد. |

## أمثلة على استخدام أوامر لغة DML:

فى هذه الفقرة سنقوم بعرض أمثلة مختلفة على استخدام الأوامر المذكورة فى الجدول السابق ، وسنستخدم فى ذلك برنامج Query Analyzer وهو أحد برامج برنامج إدارة قواعد البيانات MS SQL SERVER ، وهذا البرنامج (Query Analyzer) يستخدم فى إصدار أوامر لغة SQL إلى قاعدة البيانات ثم رؤية نتائج تنفيذ هذة الأوامر.

وسنستخدم فى الأمثلة قاعدة البيانات Northwind وهى قاعدة بيانات تأتى مدمجة مع برنامج إدارة قواعد البيانات MS SQL SERVER وتمثل Northwind قاعدة بيانات لشركة وهمية تبيع منتجات غذائية.

**ملحوظة:**

يجب أن يكون برنامج إدارة قواعد البيانات مايكروسوفت SQL سيرفر Microsoft SQL Server قد تم إعداده على جهازك.

قم بتشغيل برنامج الـ Query Analyzer وذلك عن طريق

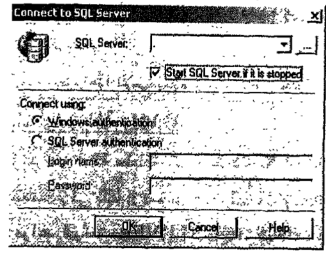
Start→Programs→Microsoft SQL Server→Query Analyzer

فتظهر نافذة كما هو واضح فى الشكل 2 ، فقم بجعل الاختيارات كما هو واضح فى الشكل 2.

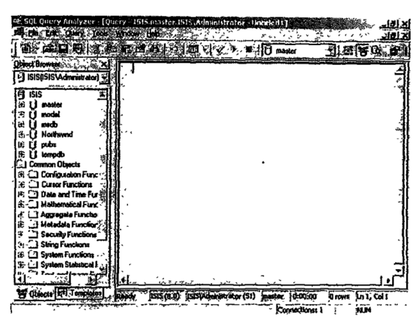
بعد ذلك تظهر نافذة برنامج Query Analyzer كما هو واضح فى الشكل 3.

ملحوظة:

يجب وضع نقطة "." فى صندوق كتابة SQL Server وهذة النقطة معناها أنها تشير إلي اسم الجهاز الخادم Server الحالى وذلك كما هو واضح فى الشكل 2.




شكل 2 فتح البرنامج




شكل 3 فتح البرنامج

## الأمر SELECT:

والأمر SELECT يستخدم فى تنفيذ الاستعلام. والاستعلام هـو طلـب سـجل Record أو مجموعة من السجلات من جدول أو أكثر مـن قاعـدة البيانـات. والأمر SELECT هو أكثر الأوامر استعمالاً فى لغة SQL وله أشكال وملحقات كثيرة ونحن هنا نقوم بعرض بسيط فقط لشكل من أشـكاله ولـو أردت معرفـة المزيد يمكنك الرجوع لكتاب متخصص فى لغة SQL.

وبناء الأمر كالتالى

**SELECT** fieldname1,fieldname2,... **FROM** tablename

حيث:

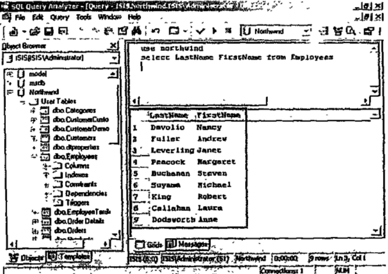
fieldname هو اسم الحقل الذى تود الاستعلام عنه ورؤيته.

tablename هو اسم الجدول الذى تستعلم فيه ويحتوى الحقول المستعلم عنها.

وكمثال قم بكتابة الأمر التالى فى نافذة الـ Query Analyzer.

```
USE northwind
SELECT LastName,FirstName from Employees
```

ثم قم بضغط الزر F5 لتنفيذ الأوامر فتجد أنه تم تنفيذ الأوامر وتم عرضها فـى جدول أسفل النافذة كما هو واضح فى الشكل 4 من المستطيل الأحمر.




شكل 4 الأمر SELECT

## شرح الكود:

في السطر الأول نخبر برنامج إدارة قواعد البيانات SQL SERVER أننا نريد العمل مع قاعدة البيانات Northwind.

في السطر الثانى نقوم باسترجاع حقلى الاسم الأول والأخير من جدول الموظفين حيث يتم كتابة أسماء الحقول التى تود استعادتها بعد الأمر select ثم نكتب بعد أسماء الحقول المفتاح from ثم اسم الجدول الموجود به الحقول.

والسؤال ماذا لو أردنا أن نقوم باسترجاع جميع الحقول الموجودة فى الجدول؟

والإجابة هى أن نستخدم المؤثر * الذى معناه "الكل" وذلك كما فى الأمر التالى

**SELECT * FROM** EMPLOYEES

وقد يتبادر إلى ذهنك عزيزى القارئ سؤال وهو كيف نستعيد سجلات بناءاً على شرط محدد؟

والإجابة هى أن نستخدم المفتاح where وذلك كما فى المثال التالى الذى نسترجع فيه سجل الموظفين الذين اسمهم الأول هو "Robert".

```
USE northwind
SELECT * FROM Employees
```

**WHERE** FirstName='ROBERT'

معنى هذا الكود أننا نريد استعادة جميع الموظفين الذين اسمهم الأول هو "ROBERT" ، حيث نكتب بعد المفتاح Where الشرط الذى نود استعادة البيانات على أساسه ، وفى هذا المثال كان الشرط هو أن الاسم الأول للموظف هو "Robert".

## الأمر INSERT:

هذا الأمر كما قلنا سابقاً يستخدم فى إدخال سجلات جديدة إلى جدول محدد وصيغة بنائه كالتالى:

**INSERT INTO** tablename (fieldname1,fieldname2, ..., fieldnameN) **Values** (value1,value2,.......,valueN)

حيث:

tablename هو اسم الجدول الذى تود إضافة سجل جديد فيه.

fieldname هو اسم الحقل الذى تود إدخال قيم فيه.

value هى القيم التى تود إدخالها فى الحقول.

لاحظ أن كلمة VALUES هى جزء من الأمر INSERT.

**مثال:**

قم بالذهاب إلى نافذة الـ Query Analyzer ثم قم بمسح أى أوامر من النافذة ثم قم بكتابة الأوامر التالية

**USE** northwind
**INSERT INTO** employees (lastname,Firstname) **VALUES** ('Hessien','Kadry')

ثم قم بالضغط على الزر F5 لتنفيذ الأوامر.

## شرح الكود:

فى هذا المثال نقوم بإضافة سجل جديد إلى الجدول EMPLOYEES ، وفى هذا السجل سنقوم بإدخال قيم للحقلين الاسم الأول والاسم الأخير فقط. وكما تلاحظ عزيزى القارئ فقد استخدمنا الأمر INSERT INTO ويليه اسم الجدول الذى نود إضافة سجل جديد فيه وهو EMPLOYEES ثم حددنا أسماء الحقول التى نود إدخال قيم فيها ثم يليها كلمة VALUES ثم القيم ('Hessien','Kadry') التى نود إدخالها.

ملحوظة:

يجب أن يكون نوع القيم التى يتم إدخالها متوافقة مع نوع بيانات الحقل وإلا سيقوم المترجم بإصدار رسالة خطأ ، فمثلاً إذا كان نوع بيان الحقل "تاريخ" فيجب إدخال قيم من نوع تاريخ وهكذا....

## الأمر Update:

يستخدم الأمر Update فى تعديل بيانات موجودة بالفعل فى جدول ما ، وأبسط شكل لهذا الأمر هو:

```
UPDATE tablename
SET
fieldname1=value1,fieldname=value2,....fieldnameN=ValueN
Where criteria
```

حيث:

tablename هو اسم الجدول الذى ترغب فى تعديل بياناته.

fieldname هو اسم الحقل الذى ترغب فى تعديل محتوياته.

value هى القيمة التى ترغب فى وضعها محل القيمة القديمة.

## وكمثال:

قم بالذهاب إلى نافذة الـ Query Analyzer ثم قم بمسح أى أوامر من النافذة ثم قم بكتابة الأوامر التالية:

```
USE Northwind
SELECT * From employees
UPDATE Employees
SET LastName='Maged',FirstName='Mostafa'
WHERE Firstname='kadry'
SELECT * From employees
```

## شرح الكود:

- السطر الأول يخبر برنامج MS SQL SERVER بأننا نريد العمل مع قاعدة البيانات Northwind.
- السطر الثانى نستخدمه فى استعادة وعرض البيانات قبل عملية التعديل.
- السطر الثالث وفيه نستخدم الأمر UPDATE حيث نقوم بتعديل السجلات التى يكون فيها الاسم الأول هو Kadry (وهو سجل واحد ناتج عن جملة الـ INSERT الموجودة فى المثال السابق) فقمنا بتعديل الاسم الأخير LASTNAME ووضعنا مكانه القيمة الجديدة Maged ثم الاسم الأول FirstName ووضعنا مكانه Mostafa.
- جملة SELECT الموجودة فى السطر الأخير موجودة لاستعادة وعرض البيانات بعد التعديل.

## الأمر Delete:

يستخدم الأمر DELETE لحذف بيانات (سجل record أو سجلات) من جدول ما وصيغته العامة كالتالى:

```
DELETE FROM tablename
WHERE criteria
```

**وكمثال:**

قم بالذهاب إلى نافذة الـ Query Analyzer ثم قم بمسح أى أوامر من النافـذة ثم قم بكتابة الأوامر التالية:

```
USE Northwind
DELETE FROM employees
WHERE Firstname='Mostafa'AND LastName='Maged'
SELECT * From employees
```

**شرح الكود:**

فى هذا المثال نستخدم الأمر DELETE لحذف السجل الذى عدلناه فى المثـال السابق ، والأمر DELETE بسيط جداً فى استخدامه حيث يـأتى بعـد كلمـة FROM اسم الجدول الذى تود حذف سجل منه ثم يأتى بعد كلمـة WHERE شرط لابد أن يتحقق لتتم عملية الحذف ، وفى هذا المثال أعطيناه شرط أن الاسم الأول هو Mostafa والاسم الأخير هو Maged. أما جملـة الـ SELECT الموجودة فى السطر الأخير فهي لعرض بيانات الجدول بعد حذف السجل منه.

## العلاقة بين لغة SQL ولغة VB.net و ADO.net:

تخيل عزيزى القارئ شخصين يتحدثان أحدهما يتكلم العربيـة والأخـر يـتكلم الألمانية وكلاهما لا يفهم لغة الأخر ، فهل يفهم أحدهما الأخر عندما يتحـدثان؟ بالطبع لا. ولكى يفهم أحدهما الأخر فلابد من وجود وسيط بينهما يفهم لغة كـل منهما وينقل محتوى كلام الشخص الأول للثانى والعكس ، هذا المثـال بالضـبط يحاكى الواقع عند لغات البرمجة فالبرنامج الذى تكتبه يتكلم لغة VB.net بينمـا برنامج قواعد البيانات الذى تتعامل معة يتكلم لغة الـ SQL وبـالطبع لا يفهـم أحدهما الأخر لذلك كان لابد من وسيط بينهما وهذا الوسيط هو الـ ADO.net.

والـ ADO.net عبارة عن **فصائل** تفهمها لغة VB.net وفى نفس الوقت لها القدرة على التعامل مع برامج قواعد البيانات والعمل بلغة SQL.

## :ADO.net

الـ ADO.net كما قلنا سابقاً أنها عبارة عن فصائل مهمتها التواصل مع برامج قواعد البيانات ، والـ ADO.net هى تطوير للـ ADO ( ActiveX Data Object) وهى التقنية التى كانت مستخدمة قبل ظهور منصة الدوت نت Net. Platform.

ومن أهم نقاط التطوير الموجودة فى الـ ADO.net هى القدرة على العمل مع قواعد البيانات وأنت غير متصل بها أو ما يعرف بـ Disconnected Data ، فالبرنامج يحتاج للاتصال بقاعدة البيانات فقط عند بداية العمل (ويتم فيها تحميل الجداول والبيانات فى فصائل الـ ADO.net) ثم يتم الانفصال عن قاعدة البيانات وتتم عمليات معالجة البيانات بشكل طبيعى جداً ، ثم إذا أردت أن يتم نقل ما تم تحديثه فى البيانات الموجودة فى الفصائل ، فنقوم مرة أخرى بالاتصال بقاعدة البيانات ويتم تحديثها ، وهذا الأسلوب يؤدى إلى سرعة فى الأداء وتقليل العبء على الشبكة وخادم قواعد البيانات Database Server.

وفصائل الـ ADO.net فى منصة الـ net Platform. موجودة فى مكتبة رئيسية هى **System.Data** ثم يتم بعد ذلك تقسيمهم إلى مكتبتين NameSpace فرعيتين هما:

- System.Data.SqlClient
- System.Data.OleDb

والسؤال الذى يتبادر إلى ذهنك عزيزى القارئ هو ما الفرق بينهما؟

والإجابة هى أن المكتبة **System.Data.SqlClient** مخصصة للتعامل مع برنامج إدارة قواعد البيانات Microsoft SQL Server فقط ، وهى أعلى أداءً وتكاملاً مع منصة الدوت نت عن مكتبة الفصائل الأخرى.

أما المكتبة **System.Data.OleDb** فهى مخصصة للتعامل مع أى نوع من أنواع قواعد البيانات بما فيها برنامج Microsoft SQL Server ، ولكنها أقل نسبياً فى الأداء والسرعة عن المكتبة الأولى.

والخبر السار بخصوص هاتين المكتبتين أن فصائلهما متشابهة فى الأسماء وطريقة العمل تقريباً وبذلك لن تحتاج عزيزى القارئ إلى مجهود كبير فى التعامل مع كليهما.

## طريقة العمل بالـ ADO.net:

لكتابة برنامج يستخدم فصائل الـ ADO.net هناك طريقتان:

- الأولى: أن نجعل بيئة فيجوال ستوديو دوت نت Visual Studio .NET تقوم بكتابة معظم الكود وهو الطريق السهل ولكن يعيبه أن هناك الكثير من الكود الذى يبدو للوهلة الأولى غير مفهوم.
- الثانية: أن نقوم بكتابة الكود بأيدينا وفيه يكون الكود أقل وأيضاً أيسر فى الفهم والتتبع وهو الطريقة التى أفضلها لك عزيزى القارئ خصوصاً عند بداية التعامل مع فصائل الـ ADO.net.

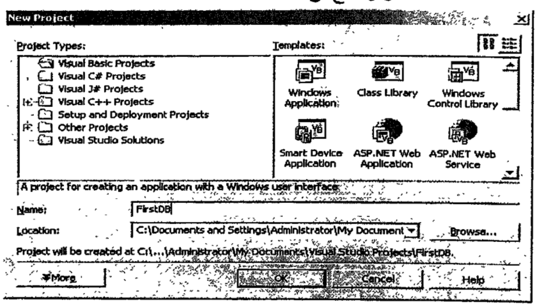
وعلى أى حال سنرى معاً الطريقتين.

### 1- الطريقة الأولى:

- قم بتشغيل بيئة فيجوال ستوديو دوت نت عن طريق

Start→Programs→ Microsoft Visual Studio .NET → Microsoft Visual Studio .NET

- قم بإنشاء مشروع جديد من نوع Windows Application وقم بتسميته FirstDB كما هو واضح في شكل 5.

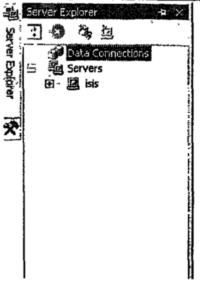



شكل 5 إنشاء المشروع

- فتجد أنه تم إنشاء مشروع وبه نموذج Form في حالة التصميم.

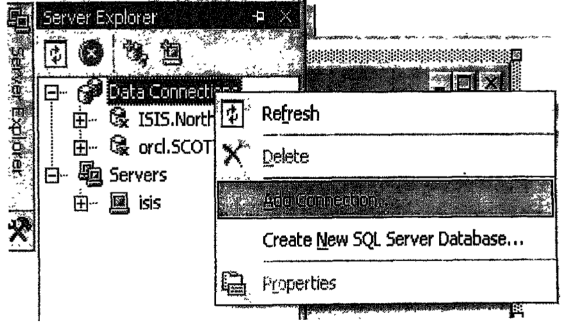
أول خطوة سنقوم بها هو تعريف وصلة مع برنامج إدارة قواعد البيانات في نافذة الـ Server Explorer واختيار قاعدة البيانات وذلك عن طريق تنفيذ الآتي:

- قم بالوقوف بالفارة Mouse على Server Explorer في أقصى شمال الشاشة (إذا لم تكن موجودة فقم بالذهاب إلى قائمة View→ Server Explorer ) وستجد نافذة جانبية قد ظهرت كما هو واضح في شكل 6.




شكل 6 تعريف وصلة مع برنامج إدارة قواعد البيانات

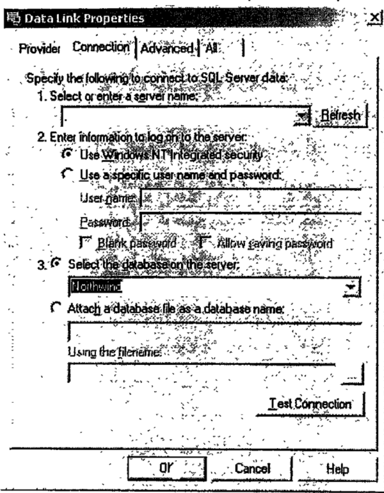
- قم بالوقوف بالفارة Mouse على Data Connections ثم اضغط بــالزر الأيمن للفــارة Mouse ثــم اختــر مــن القائمــة التــى ســتظهر Add Connection.. وذلك كما هو واضح فى الشكل 7.




شكل 7 تعريف وصلة مع برنامج إدارة قواعد البيانات

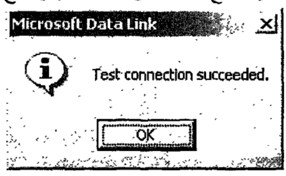
- ستجد أنه تم ظهور نافذة عنوانها Data Link properties وذلك كما هــو واضح فى الشكل 8 ، فقم بكتابة نقطة "." فى القائمة المنسدلة التى عنوانهــا

Server أو أكتب اسم الجهاز الخـادم Select or enter a server name إذا كنت تعرفه ، ثم اختـر الاختيـار Use Windows NT Integrated Security وهذا الاختيار لتعريف مستوى الأمن الذى تريده وقد اخترنا نظام الأمن المدمج فى نظام النوافذ Windows ، ثم اختر الاختيار Northwind من القائمة المنسدلةSelect the Database on the Server وهذه القـائمة كما هو واضح من اسمها تفيد فى اختيار قاعدة البيانات التى نود الاتصـال والعمل معها (انظر شكل 8).



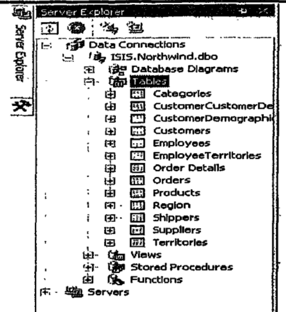

شكل 8 تعريف وصلة مع برنامج إدارة قواعد البيانات

- قم بعد ذلك بالضغط على الزر Test Connection وذلك لاختبار هل تم الاتصال مع قاعدة البيانات بنجاح أم لا ، فإذا تم الاتصال بنجاح ستجد أنه تم ظهور رسالة تفيد نجاح الاتصال وذلك كما هو واضح فى شكل 9.




شكل 9 تعريف وصلة مع برنامج إدارة قواعد البيانات

- قم بالذهاب بالفارة Mouse مرة أخرى إلى النافذة الجانبية Server Explorer وستجد أنه تمت إضافة وصلة تحتوى اسم خادم قاعدة البيانات Server Database ثم نقطة ثم اسم قاعدة البيانات Northwind ثم اسم مالك قاعدة البيانات فقم بالضغط على علامة "+" الموجودة بجانب اسم الوصلة وذلك لرؤية محتويات الوصلة وستجد هناك خمس عناصر رئيسية فقم بالضغط على علامة "+" التى بجانب العنصر Tables وذلك لرؤية الجداول المكونة لقاعدة البيانات Northwind وذلك كما هو واضح فى الشكل 10.

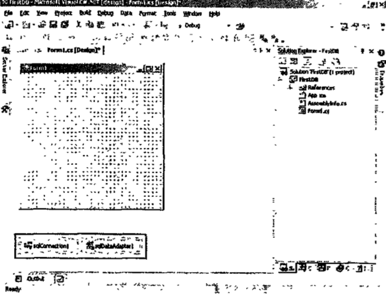



شكل 10 تعريف وصلة مع برنامج إدارة قواعد البيانات

- بذلك نكون قد انتهينا من إنشاء وصلة فى الـ Server Explorer لقاعدة البيانات التى نرغب فى العمل معها.

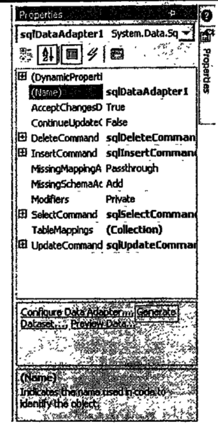
## الخطوة الثانية التالية هى الإنتقال للعمل مع لغة VB.net:

- قم بالذهاب إلى نافذة الـ Server Explorer ثم من عنصر الجداول Tables اضغط على الجدول Employees واسحبه وقم بوضعه على النموذج Form.
- ستجد أنه تمت إضافة اثنين من الأيقونات icons أسفل النموذج Form يمثلان هدفين Instances من فصيلتين من فصائل الـ ADO.net وهما SqlConnection و SqlDataAdapter كما سنوضحهم لاحقاً وذلك كما هو واضح فى الشكل 11 وقد وضحتهم بمستطيل أحمر حولهم.



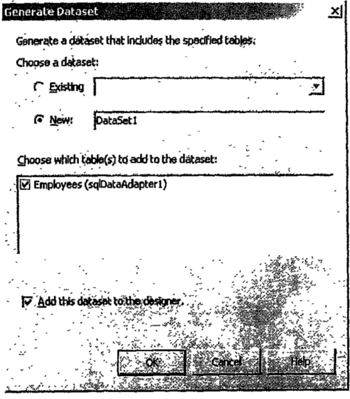
شكل 11 تعريف وصلة مع برنامج إدارة قواعد البيانات

- اضغط بالزر الأيسر للفارة Mouse على أى مكان فـى النمـوذج Form وذلك لكـى تقـوم بإزالـة حالـة الاختيـار Selection عـن الهـدفين sqlDataAdapter1 و sqlConnection1.
- ثم قم بالضـغط بـالزر الأيمـن للفـارة Mouse علـى أيقونـة الهـدف sqlDataAdapter1 ثم اختر Properties من القائمة المختصرة ليظهـر صندوق الخصائص فقم بالضغط على الوصلة Generate DataSet وذلك كما هو واضح فى الشكل 12 (الوصلة وضعنا حولها مستطيل أحمر).



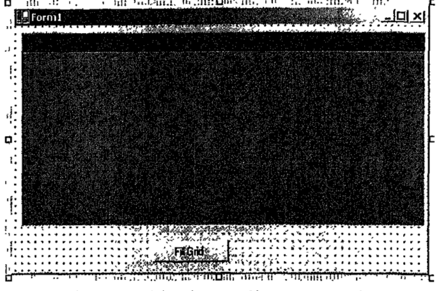

شكل 12 تعريف وصلة مع برنامج إدارة قواعد البيانات

- بعد الضغط على الوصلة تجد أنه تم ظهور نافذة عنوانها Generate DataSet فقم بترك الاختيارات كما هى ثم اضغط الزر OK وذلك كما هو واضح فى الشكل 13.




شكل 13 تعريف وصلة مع برنامج إدارة قواعد البيانات

- ستجد أنه تم إنشاء أيقونة icon ثالثة أسفل النموذج Form عنوانها dataSet11 وهي تمثل هدف Instance من الفصيلة DataSet.
- قم بعد ذلك بسحب أداة DataGrid و Button من نافذة صندوق الأدوات ToolBox من يسار الشاشة وضعهما على النموذج Form وقم بتغيير أبعادهما ليبدوان كما هو واضح في الشكل 14.
- قم بتغيير عنوان الزر إلى Grid Fill وذلك من خاصية Text من خلال شاشة الخصائص Properties Window.




شكل 14 تعريف وصلة مع برنامج إدارة قواعد البيانات

- قم بالضغط مرتين متتاليتين Double-click على الزر Fill Grid وستجد أنه تم ظهور نافذة الكود مع وقوف المؤشر عند دالة معالجة حدث الضغط OnClick للزر Fill Grid فقم بكتابة الكود التالى بين الأقواس:

```
SqlDataAdapter1.Fill(DataSet11)
DataGrid1.DataSource = DataSet11.Tables(0)
```

- قم الآن بتنفيذ البرنامج عن طريق الضغط علي الزر F5 وستجد أن البرنامج يتم تنفيذه بشكل صحيح إذا اتبعت الخطوات السابقة بدون أخطاء.
- لاحظ أن أداة شبكة البيانات خالية ولايوجد بها أى بيانات فقم بالضغط على الزر Fill Grid وستجد أنه تم ملئ الأداة DataGrid1 بالبيانات الموجودة فى جدول الموظفين وذلك كما هو واضح فى شكل 15.
- حاول أن تلقى نظرة على كامل الكود الموجود عزيزى القارئ وستجد أنه تم كتابة سطور كثيرة من الكود التى سنتناول أهم ما فيها من خلال مثال نقوم فيه بكتابة الكود بأيدينا.

Form1

| | EmployeeID | LastName | FirstName | Title | TitleOfCourte | BirthD |
|---|---|---|---|---|---|---|
| ▶ | 1 | Davolio | Nancy | Sales Repres | Ms | 12/8/1 |
| | 2 | Fuller | Andrew | Vice Presiden | Dr | 2/19/1 |
| | 3 | Leverling | Janet | Sales Repres | Ms. | 8/30/1 |
| | 4 | Peacock | Margaret | Sales Repres | Mrs. | 9/19/1 |
| | 5 | Buchanan | Steven | Sales Manag | Mr. | 3/4/19 |
| | 6 | Suyama | Michael | Sales Repres | Mr. | 7/2/19 |
| | 7 | King | Robert | Sales Repres | Mr. | 5/29/1 |
| | 8 | Callahan | Laura | Inside Sales | Ms | 1/9/19 |
| | 9 | Dodsworth | Anne | Sales Repres | Ms. | 1/27/1 |

Fill Grid

شكل 15 تنفيذ التطبيق

## الطريقة الثانية:

إذا أردت أن تبنى تطبيق قواعد بيانات باستخدام تقنية الـــ ADO.net فهنــاك بعض الفصائل الأساسية التى يجب أن تستخدمها وهى:

| اسم الفصيلة | عمل الفصيلة |
|---|---|
| SqlConnection | مهمتها عمل اتصال بين البرنامج الذى تبنيه وبرنامج إدارة قواعد البيانات SQL Server وهى بمثابة كوبرى تستخدمه فصائل أخرى فى عملها مع قواعد البيانات. |
| SqlCommand | مهمة هذه الفصيلة تنفيذ جمل الـ SQL داخل برنامج إدارة قواعد البيانات SQL Server ولابد أن تستخدم هدفاً object من نوع SqlConnection لأداء عملها. |
| SqlDataAdapter | تعمل هذه الفصيلة وكأنها شاحنة تتحرك بين البرنامج الذى تبنيه وبرنامج إدارة قواعد البيانات SQL Server فهى تقوم بإحضار البيانات الناتجة عن تنفيذ |

| | |
|---|---|
| | جمل SQL ثم تعود للبرنامج وتقوم بملئ هدف object من نوع DataSet ، وهى فى عملها تستخدم أهدافاً objects من نوعى SqlConnection و SqlCommand. |
| **DataSet** | هذه الفصيلة تعمل كمخزن يتم فيه تخزين جداول بأكملها أو جزء من جداول بل ويمكن تخزين قاعدة بيانات بأكملها داخل هذه الفصيلة class ، ويتم ملئ هذه الفصيلة بالبيانات والجداول عن طريق استخدام الفصيلة **SqlDataAdapter**. |

ولكى نصيغ ما جاء فى الجدول السابق فإنه لكى تقوم بإنشاء تطبيق قواعد بيانات ، فقم باتباع الخطوات التالية:

1. قم بإنشاء وصلة connection مع قاعدة البيانات عن طريق إنشاء هدف من نوع الفصيلة SqlConnection.
2. قم بإنشاء أمر يستطيع تنفيذ جمل SQL داخل برنامج إدارة قواعد البيانـــات وذلك عن طريق إنشاء هدف من نوع الفصيلة SqlCommand.
3. قم بإنشاء هدف من نوع الفصيلة SqlDataAdapter وهى كما قلنا بمثابـــة الشاحنة التى يتم ملؤها بالبيانات والجداول الناتجة عن تنفيذ جملة SQL ثـــم تعبر إلى البرنامج محملة بهذة البيانات.
4. نقوم بإنشاء هدف من نوع فصيلة DataSet وهى تعتبر بمثابة المخزن الذى يتم تعبئته بالبيانات والجداول الموجودة فى الـــ SqlDataAdapter.

فتعال معى عزيزى لنطبق الخطوات الأربع بشكل عملى.

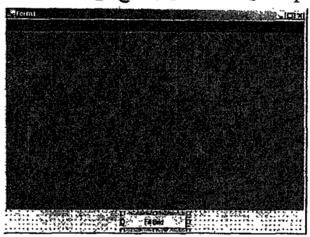
قم بإغلاق المشروع الحالى وذلك عن طريق القائمة

File→Close Solution

قم بإنشاء مشروع جديد وقم بتسميته MyDB.

قم بسحب أداة زر Button وأداة DataGrid من صندوق الأدوات Toolbox وضعهما على النموذج Form واجعل أبعادهما تبدو كما هو واضح في شكل 16.

قم بتغيير عنوان الزر إلى Fill Grid وذلك من خاصية Text وأيضاً قم بتغيير اسمه إلى FillGrid من خاصية Name وذلك من شاشة الخصائص Properties Window وذلك كما هو واضح في شكل 16.



شكل 16 تطبيق قواعد البيانات Database

قم بالضغط مرتين متتابعتين Double-click على الزر Fill Grid لتجد أنه تم نقلك إلى نافذة كتابة الكود فقم بكتابة الكود المظلل بنفس ترتيبة التالي:

```
1. Imports System.Data
2. Imports System.Data.SqlClient
3. Public Class Form1
4. Inherits System.Windows.Forms.Form
5. Private con As SqlConnection
6. Private cmd As SqlCommand
7. Private dtaAdptr As SqlDataAdapter
8. Private ds As DataSet
9. #Region " Windows Form Designer generated code "
```

```
10. Public Sub New()
11. MyBase.New()
12. 'This call is required by the Windows Form Designer.
13. InitializeComponent()
14. InitializeDB()
15. 'Add any initialization after the InitializeComponent()
    call
16. End Sub
17. 'Form overrides dispose to clean up the component
    list.
18. Protected Overloads Overrides Sub Dispose(ByVal
    disposing As Boolean)
19. If disposing Then
20. If Not (components Is Nothing) Then
21. components.Dispose()
22. End If
23. End If
24. MyBase.Dispose(disposing)
25. End Sub
26. 'Required by the Windows Form Designer
27. Private components As
    System.ComponentModel.IContainer
28. 'NOTE: The following procedure is required by the
    Windows Form Designer
29. 'It can be modified using the Windows Form
    Designer.
30. 'Do not modify it using the code editor.
31. Friend WithEvents Button1 As
    System.Windows.Forms.Button
32. Friend WithEvents DataGrid1 As
    System.Windows.Forms.DataGrid
33. <System.Diagnostics.DebuggerStepThrough()>
    Private Sub InitializeComponent()
```

```
34. Me.Button1 = New System.Windows.Forms.Button
35. Me.DataGrid1 = New
    System.Windows.Forms.DataGrid
36. CType(Me.DataGrid1,
    System.ComponentModel.ISupportInitialize).BeginIni
    t()
37. Me.SuspendLayout()
38. '
39. 'Button1
40. '
41. Me.Button1.Location = New
    System.Drawing.Point(96, 240)
42. Me.Button1.Name = "Button1"
43. Me.Button1.Size = New System.Drawing.Size(56, 24)
44. Me.Button1.TabIndex = 0
45. Me.Button1.Text = "Fill Grid"
46. '
47. 'DataGrid1
48. '
49. Me.DataGrid1.DataMember = ""
50. Me.DataGrid1.HeaderForeColor =
    System.Drawing.SystemColors.ControlText
51. Me.DataGrid1.Location = New
    System.Drawing.Point(0, 0)
52. Me.DataGrid1.Name = "DataGrid1"
53. Me.DataGrid1.Size = New System.Drawing.Size(288,
    232)
54. Me.DataGrid1.TabIndex = 1
55. '
56. 'Form1
57. '
58. Me.AutoScaleBaseSize = New
    System.Drawing.Size(5, 13)
```

```
59. Me.ClientSize = New System.Drawing.Size(292, 273)
60. Me.Controls.Add(Me.DataGrid1)
61. Me.Controls.Add(Me.Button1)
62. Me.Name = "Form1"
63. Me.Text = "Form1"
64. CType(Me.DataGrid1,
    System.ComponentModel.ISupportInitialize).EndInit(
    )
65. Me.ResumeLayout(False)
66. End Sub
67. #End Region
68. Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click
69. Try
70. con.Open()
71. dtaAdptr.SelectCommand = cmd
72. dtaAdptr.Fill(ds, "Employees")
73. con.Close()
74. DataGrid1.DataSource = ds
75. DataGrid1.DataMember = "Employees"
76. Catch ex As Exception
77. MsgBox(ex.Message)
78. Finally
79. con.Close()
80. End Try
81. End Sub
82. Private Sub InitializeDB()
83. con = New
    SqlConnection("server=.;uid=sa;pwd=123;database=n
    orthwind")
84. cmd = New SqlCommand("select * from
    employees",con)
```

```
85. dtaAdptr = New SqlDataAdapter
86. ds = New DataSet
87. End Sub
88. End Class
```

## شرح الكود:

كما ترى فإن هذا الكود يماثل ما هو موجود عندك باستثناء المظلل منه وهو الذى أضفناه فقم بإضافته للكود عندك وسنقوم بشرحه حالاً.

- كما تعرف عزيزى القارئ أنه لكى نقوم بإنشاء هدف object يمكن أن يتم ذلك بطريقتين: إما على مرحلة واحدة أو على مرحلتين كالتالى:

Dim g as New Dog()

هذا الشكل يمثل إنشاء هدف Object على مرحلة واحدة حيث Dog هى فصيلة ما ، أما g فهو اسم الهدف object.

أو

Dog g
G=new Dog()

هذا الشكل يمثل إنشاء هدف object على مرحلتين حيث:

تعتبر g فى الخطوة الأولى مؤشراً Reference وليست هدفاً Object.

أما فى الخطوة الثانية فقد أنشأنا هدفاً object فى الذاكرة وأصبح المؤشر g يشير له وهذا هو الأسلوب الغالب فى إنشاء الأهداف object وذلك كما سترى من خلال الكود السابق وسنوضحه فى الشرح.

- فى السطر 1و2 قمنا بالإشارة إلى المكتبات التى توجد فيها فصائل الـADO.net والتى يمكنها العمل مع برنامج SQL Server.
- فى السطر 5 قمنا بإنشاء مؤشر اسمة con من نوع الفصيلة SqlConnection وهى المسئولة عن إنشاء وصلة مع قاعدة البيانات.

- وفى السطر 6 قمنـا بإنشـاء مؤشـر اسـمه cmd مـن نـوع الفصـيلة SqlCommand وهى الفصيلة المسئولة عن تنفيذ أوامـر داخـل SQL Server .
- فى السطر 7 قمنا بإنشاء مؤشر اسـمه dtaAdptr مـن نـوع الفصـيلة SqlDataAdapter.
- فى السطر 8 قمنا بإنشاء مؤشر اسمه ds من نوع الفصيلة DataSet.
- فى السطور من 81 إلى 87 قمنا بإنشاء الدالة InitializeDb() وهى دالـة سوف تكون مهمتها إنشاء أهداف للمؤشرات السابق الإعـلان عنهـا فـى السطور من 5-8.
- فى السطر 82 نقوم بإنشاء هدف Object من الفصـيلة SqlConnection ونربطه مع المؤشر con كالأتى:

```
con=new SqlConnection()
```

ولكن عند إنشاء وصلة مع قاعدة البيانات لابد من إعطاء الوصلة معلومات هى:

1- اسم الجهاز الخادم Server الموجود به قاعدة البيانات ويرمز لها باسم Server.
2- اسم المستخدم ويرمز له بالاسم uid.
3- كلمة السر الخاصة به ويرمز لها بالاسم pwd.
4- اسم قاعدة البيانات التى نرغب فى العمل معها ويرمـز لهـا بالاسـم database.

ويتم تمرير هذه المعلومات من خـلال دالـة البنـاء Constructor كمعامـل parameter حرفى وذلك كالتالى:

```
con=new SqlConnection("server=.;uid=sa;pwd=123; database=
Northwind");
```

حيث أن اسم الجهاز الخادم Server "." وهو يشير إلى اسم الجهاز الخادم Server الحالى ولو كنت تعمل من خلال شبكة يمكنك كتابة إسم الخادم الموجود به قاعدة البيانات.

واسم المستخدم sa يشير إلى مدير قاعدة البيانات "System Administrator".

وكلمة السر 123 ولاحظ أنك يجب أن تكتب كلمة السر الخاصة بك كما هى موجودة فى إعدادات جهازك.

واسم قاعدة البيانات التى نرغب العمل معها هى Northwind.

لاحظ أنه لابد أن يفصل بين كل معلومة فاصلة منقوطة ";".

- فى السطر 83 نقوم بإنشاء هدف من نوع SqlCommand وربطه بالمؤشر cmd السابق الإعلان عنه فى السطر 6 ، ولكى ننشئ هدفاً object من هذه الفصيلة class ، فيجب تمرير passing معاملين parameters لها من خلال دالة البناء Constructor هما:

1. المعامل الأول جملة الـ SQL التى سينفذها هذا الهدف object ، وفى حالتنا هذه تجد أننا مررنا له الجملة

"Select * from employees"

2. المعامل الثانى يجب أن يكون هدفاً object من نوع SqlConnection (الكوبرى الذى من خلاله يستطيع الوصول إلى قاعدة البيانات) وستجد أننا فى حالتنا هذه مررنا لة الهدف con.

- فى السطر 83 نقوم بإنشاء الهدف dtaAdptr من نوع SqlDataAdapter وهى كما قلنا تعمل بمثابة شاحنة تحمل البيانات الناتجة عن تنفيذ جملة الـ SQL الموجودة فى الهدف cmd.

- فى السطر 85 نقوم بإنشاء هدف ds من نوع DataSet وهو كما قلنا يعمل بمثابة المخزن الموجود فى برنامجك ويتم ملء هذا المخزن عن طريق الهدف (الشاحنة) dtaAdptr.
- فى السطر 14 يتم استدعاء Calling الدالة InitializeDb() من داخل دالة البناء للنموذج Form Constructor وذلك حتى يتم تنفيذ هذه الدالة Method عند بداية تحميل النموذج Form فتصبح الأهداف objects جاهزة للاستعمال.
- فى السطور من 69-79 نجدها مكتوبة فى دالة الحدث onClick للزر Fill Grid وفيها نجد الأتى:
- تمت إحاطة الجمل بمعالج الاستثناء try..catch وذلك لأنه قد يحدث استثناء Exception مثل أن يكون خادم قاعدة البيانات Database Server مغلقاً أو أن قاعدة البيانات غير موجودة أو أنة لايوجد جدول بالاسم الذى تحاول التعامل معه.
- فى السطر 70 نقوم بفتح الوصلة مع قاعدة البيانات وذلك عن طريق استدعاء الدالة con.Open() وهذه خطوة ضرورية جداً حيث لابد قبل أن تقوم بأى عمليات فى قاعدة البيانات ، أن تفتح الوصلة (الهدف con كما قلنا يعد فقط بمثابة كوبرى مغلق يتم فتحه باستخدام الدالة Open()).
- فى السطر 71 نخبر الهدف dtaAdptr بالهدف الذى ينفذ العمليات (فى هذه الحالة فإن العملية من نوع select ومن الممكن أن تكون العملية من نوع Insert فهنا نستخدم الأمر Insert أو أى أمر أخر من أوامر لغة DML التى تحدثنا عنها فى بداية هذا الفصل) داخل قاعدة البيانات وهذا الهدف بالطبع هو الهدف cmd وذلك عن طريق الجملة التالية:

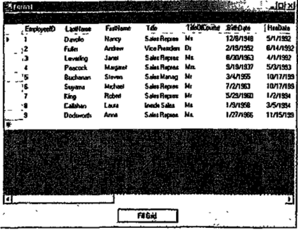
```
dtaAdptr.SelectCommand=cmd;
```

- فى السطر 72 نقوم بملء الهدف( المخزن) ds بالبيانات وذلك عن طريق استدعاء الدالة ()dtaAdptr.Fill وهذه الدالة تستقبل معاملين Parameters هما:

أ- هدف object من نوع DataSet أى المخزن وستجد أننا مررنا له الهدف ds.

ب- عند ملء هذا الهدف بالبيانات ، تكون البيانات فى شكل جدول table ولذلك يفضل أن تعطى اسماً لهذا الجدول الموجود فى المخزن وأحياناً نعطيه نفس الاسم الموجود فى قاعدة البيانات وستجد فى حالتنا هذه أننا أعطيناه الاسم "Employees" ، وبهذا أصبحت محتويات الجدول Employees الموجود فى قاعدة البيانات Northwind موجودة لدينا فى الهدف ds ونستطيع التعامل مع تلك البيانات ونحن غير متصلين بقاعدة البيانات ، وبذلك لم نعد فى حاجة للاستمرار فى الاتصال بقاعدة البيانات.

- فى السطر 73 نقوم بإغلاق الاتصال بقاعدة البيانات وذلك عن طريق استدعاء الدالة ()con.Close وهذه خطوة هامة إذ يجب عليك كما فتحت قناة اتصال مع قاعدة بيانات ، أن تغلقها فور الانتهاء من العمل معها.
- فى السطر 74 نخبر الهدف DataGrid1 الذى يمثل الأداة DataGrid أن مصدر البيانات هو الهدف ds الذى يعتبر بمثابة قاعدة البيانات.
- فى السطر 75 نخبر الهدف dataGrid1 أن يأخذ البيانات من الجدول "Employees" الموجود فى الهدف ds.
- قم الآن بتنفيذ البرنامج وستجد أن الأداة DataGrid فارغة ، فقم الآن بالضغط على الزر Fill Grid وستجد أن الأداة قد امتلأت بالبيانات وذلك كما هو واضح فى الشكل 17.



| EmployeeID | LastName | FirstName | Title | TitleOfCourtesy | BirthDate | HireDate |
|---|---|---|---|---|---|---|
| 1 | Davolio | Nancy | Sales Repres | Ms | 12/8/1948 | 5/1/1992 |
| 2 | Fuller | Andrew | Vice Presiden | Dr | 2/19/1952 | 8/14/1992 |
| 3 | Leverling | Janet | Sales Repres | Ms | 8/30/1963 | 4/1/1992 |
| 4 | Peacock | Margaret | Sales Repres | Mrs | 9/19/1937 | 5/3/1993 |
| 5 | Buchanan | Steven | Sales Manag | Mr | 3/4/1955 | 10/17/199 |
| 6 | Suyama | Michael | Sales Repres | Mr | 7/2/1963 | 10/17/199 |
| 7 | King | Robert | Sales Repres | Mr | 5/29/1960 | 1/2/1994 |
| 8 | Callahan | Laura | Inside Sales | Ms | 1/9/1958 | 3/5/1994 |
| 9 | Dodsworth | Anne | Sales Repres | Ms | 1/27/1966 | 11/15/199 |



شكل 17 تنفيذ التطبيق

حاول فتح المشروع السابق وحاول تتبع الكود وستجد أن معظم الكود يماثل ماقمنا به فى هذا المثال ولكن مع الاختلاف أنه قد استخدم أكثر من هدف object من نوع SqlCommand وذلك لأنه يفترض أنك تريد القيام بعمليات SQL مختلفة مثل الإضافة والحذف والتعديل ، وأخيراً أود أن أقول أن ما تعرضنا له خلال هذا الفصل يعد بمثابة خطوة أولى فى طريق طويل عنوانه برمجة قواعد البيانات ، وإذا أردت أن تعرف الكثير عن برمجة قواعد البيانات فانتظر كتاب VB.NET وبرمجة قواعد البيانات.

# الفصل الثاني عشر

## صفحات الخادم النشطة

## Active Server Pages (ASP .NET)

صفحات الخادم النشطة
Active Server Pages (ASP .NET)

فى هذا الفصل سوف نتعرف علي كيفية التعامـل مـع قواعد البيانات Databases علـي شـبكة الإنترنـت Internet في لغة VB.NET من خلال تقنيـة ASP .NET. وذلك من خلال النقاط التالية:

1. مقدمة Introduction.
2. ميكانيكية عمل الويب Web.
3. معمارية نظم الويب Web.
4. الفرق بين الصفحات الساكنة والديناميكية Static and Dynamic web Pages.
5. إنشاء صفحات الـ Asp.net.
6. استخدام أدوات الويب Web Controls.
7. تطبيق عملى باستخدام ASP .NET.

## مقدمة:

تعلمنا فى الفصول السابقة كيفية استخدام نمـــاذج النوافــذ Windows Forms واستخدام الأدوات المصاحبة لها Windows Controls لنقوم ببنـــاء تطبيقـــات موجهة للعمل فى بيئة ويندوز Windows Platforms Applications. وفـــى هذا الفصل سوف نقوم بمعرفة كيفية بناء تطبيقات موجهة للعمل من خلال شبكة الإنترنت Web-based applications باستخدام تقنية الـــ Asp.net.

ولكن قبل التعرض لكيفية بناء تطبيقات الويب Web Applications ، فلابد من معرفة ميكانيكية عمل شبكة الإنترنت Internet وذلك من خلال الفقرة التالية.

## ميكانيكية عمل الويب Web:

من المعروف أن الإنترنت ما هى إلا مجموعة كبيرة من الشبكات حول العـــالم ، وهذه الشبكات مرتبطة ببعضها البعض بحيث تكون –فيما يمكن قولـــه مجـــازاً– شبكة واحدة كبيرة جداً. والشبكة ما هى إلا مجموعة من أجهزة الكمبيوتر متصلة ببعضها البعض بغرض تبادل المعلومات وخدمات أخرى. ولكى تســـتطيع هـــذه الأجهزة فهم بعضها ، فلابد أن يكون هناك لغة تفهمها جميع الأجهزة الموجـــودة فى الشبكة وهذه اللغة تسمى فى مجال الشبكات بالبروتوكول Protocols. وأهم برتوكول Protocol موجود فى شبكة الأنترنت هو TCP/IP. ويوجد بروتوكول Protocol أخر يسمى HTTP ويعمل مـــن خـــلال البروتوكـــول TCP/IP ، والبروتوكول HTTP هو الذى يستخدم فى نقل صفحات الويـــب Web Pages من خادم الويب Web Server (خـــادم الويـــب Web Server هـــو برنـــامج Software موجود فى الجهاز الخادم Server على الإنترنت Internet) إلـــى المتصفح Browser.

ومن المهم معرفة كيفية نقل الصفحات ولذلك لابد من نظرة سريعة على العناصر المكونة لهذه العملية وهى:

1. عميل Client ويعنى جهاز كمبيوتر كما هو لديك وهــذا الجهــاز مــزود ببرنامج متصفح Browser (أحياناً يطلق على المتصفح أنه Client) مثــل برنامج الإنترنت إكسبلورر Internet Explorer أو نتســكيب Netscape Navigator.
2. شبكة الإنترنت Internet.
3. خادم ويب Web Server وهو جهاز كمبيوتر مزود ببرنامج مهمته تلقــى طلبات requests ثم إرسال استجابات Responses ، وهــذا الجهــاز ذو إمكانيات كبيرة ويعمل على مدار اليوم وهو الذى يحتوى صــفحات الويــب Web Pages وأحياناً يسمى العائــل أو المستضــيف Host وذلــك لأنــه يستضيف مواقع وصفحات الويب Web Pages.

فمثلاً أنت عندما تكتب العنوان التالى فى متصفحك Browser:

http://www.islamonline.net/news/arabic.html

ماذا يحدث؟

لو دققنا النظر سنجد أن هذا العنوان ينقسم إلى ثلاثة أقسام هم:

1. //:http وهو اسم البروتوكول Protocol المستخدم فى عملية الحصول على الصفحات.
2. www.islamonline.net هوالاسم الكامل للعائل Host أى اسم الكمبيوتر الذى يحتوى الموقع الذى تريد رؤية صفحاته ، وهذا الاسم يتم ترجمته إلى عنوان إليكتروني IP مثل (205.40.125.220) وذلك لأن كل عائل Host على الإنترنت له IP خاص به ولا يتكرر أبداً.
3. باقى العنوان /news/arabic.html يحدد اسم الصفحة resource ومكانها فى العائل Host ، ففى العنوان السابق نجد أن الصــفحة المطلوبــة هــى

Arabic.html ومكانها (مسارها path) موجود فى مجلد news فى العائل .Host

وبعد أن تعرفنا على عناصر العملية تعال معى لنرى كيف يتم التعامل بين هـذه العناصر:

1. عندما يتم كتابة عنوان ما مثل

http://www.islamonline.net/news/arabic.html

فى المتصفح Browser فإن المتصـفح Browser يقـوم بعمليـة تسـمى HTTP transaction (أى طلب نقل ملف ما بين برنـامج خـادم الويـب Web Server والمتصفح Browser) وذلك لجلب وعرض ملف مـا فـى المتصفح Browser ، وفى هذه العملية يقوم المتصفح Browser بإرسـال طلب HTTP Request إلى الجهاز الخادم Server وذلك كالتالى:

Get /news/Arabic.html HTTP/1.1

والكلمة Get هى أمر من أوامر البروتوكول HTTP (HTTP Method) وهذا الأمر يحدد رغبة العميل (المتصفح Browser) فى الحصـول علـى ملف ما من الخادم Server ، أما باقى الأمر فيحدد مسـار واسـم الملـف المرغوب في الحصول عليه وأيضاً اسـم البرتوكـول Protocol ورقـم إصداره version (HTTP /1.1).

2. يأتى هذا الطلب للخادم Server الذى يقوم بترجمة هذا الطلب ثم يقوم بتنفيذ هذا الطلب ثم يرسل للعميل Client باستجابة response وهذه الاسـتجابة تكون فى شكل نص يحتوى على البروتوكول Protocol المستخدم ثم رقـم كودى توضح حالة الاستجابة كما فى الشكل التالى

HTTP/1.1 200 OK

وهذا الشكل من الاستجابة يوضح أن الخادم Server وجد الملف المطلوب ، ثم يقوم الخادم Server أيضاً بإرسال ما يسمى بـالرأس HTTP headers

الذى يوضح معلومات عن نوع الملف الذى سيقوم بإرساله ، فمثلاً إذا كـان الخادم Server سيرسل ملفاً من نوع HTML فيكـون الـرأس Header كالتالى:

Content-type: text/html

أما لو كان الملف من نوع صورة مثلاً فيكون الرأس Header كالتالى:

Content-type: image/gif

والمتصفح Browser يستفيد من الرأس Header لأنه يستخدمه فى معرفة ما إذا كان قادراً على عرض هذا الملف أم يحتاج إلـى برنـامج خـارجى لعرضه ، ففى حالة عدم قدرته على عرض نوع الملف الذى يرسله الخـادم Server ، فهنا يقـوم المتصـفح Browser بإظهـار رسـالة الـتحميل Download.

ثم بعد إرسال الرأس Header (أو مجموعة من الرؤوس Headers) يأتى بعدها سطر فارغ ومعناه أن الخادم Server قد انتهى من إرسـال الـرأس Header ، ثم يقوم الخادم Server بعد ذلك بإرسال ذلك الملف فى صورة تدفق Html Stream.

ويأتى هذا التدفق Stream إلى المتصفح Browser الـذى يقـوم بتدقيقـه وتحليله parsing ثم يقوم بتحويله إلى ملف ويقوم بعرضه.

أما لو كانت رسالة الاستجابة كالتالى:

HTTP/1.1 404 Not Found

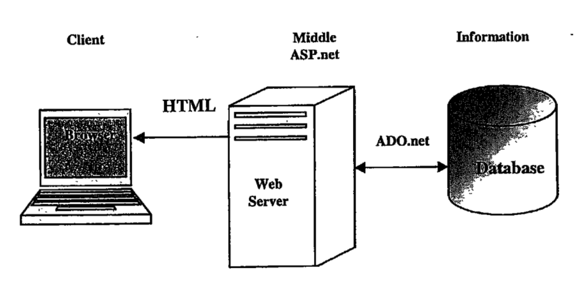
فإن هذا معناه أن الخادم Server لم يجد الملف المطلوب.

## معمارية نظم الويب System Architecture:

عادة ما تقسم التطبيقات إلى تطبيقات الجهاز الواحد Desktop Applications و تطبيقات العميل/الخادم Client/Server Applications ، فالـ Desktop Applications هى التى تعمل على جهاز واحد مثـل برنـامج الـوورد MS

WORD وإكسيل MS EXCEL أى يكون البرنامج كيان واحد على جهاز واحد فقط ، أما تطبيقات العميل/ الخادم فتسمى التطبيقات الموزعة Distributed Applications ففيها يكون التطبيق كيانات متعددة (أجزاء متعددة) فمثلاً يوجد جزء على جهاز العميل Client وجزء أخر على الخادم Server ، والعميل والخادم مرتبطين من خلال الشبكة.

وتطبيقات الويب Web تنتمي إلى الـ Distributed Application ولكن بمفهوم أوسع ، فمثلاً الـ Client/Server يكون عدد العناصر التي تحتوى البرنامج اثنين هما العميل Client والخادم Server ويطلق على العميل والخادم لفظ طبقة tier فهنا يطلق على تطبيقات الـ Client/Server تطبيقات ثنائية الطبقة Two-tier Applications ، أما تطبيقات الإنترنت فيطلق عليها N-tier Applications وذلك لوجود أكثر من طبقة tier يتكون منها التطبيق وذلك كما هو واضح فى الشكل 1.




شكل 1 تطبيقات الإنترنت Internet

وكما نرى فى الشكل فإن هناك أكثر من طبقة فهناك:

- طبقة البيانات data tier والتى تحتوى البيانات التى التى يعالجها البرنامج وهذه الطبقة يمثلها أحد برامج إدارة قواعد البيانات الارتباطية Relational Database Management Systems(RDBMS) والتى تعرضنا لها فى الفصل السابق.
- الطبقة الوسطى middle tier هى التى تتحكم فى طريقة عرض البيانات القادمة من الـ Data tier وهى تتحكم فى التفاعل بين العميل client وطبقة البيانات Data tier ، فمثلاً عندما يريد العميل رؤية المنتجات الموجودة فى قاعدة البيانات ، فإنه يقوم بإرسال طلب إلى الـ middle tier ثم تقوم الـ middle tier بإرسال الطلب إلى قاعدة البيانات فتقوم قاعدة البيانات بإرسال البيانات المطلوبة ولكن هذه البيانات تكون غير مشكلة unformatted أى تكون فى هيئة قبيحة غير مقبولة الشكل فتقوم الـ middle tier بتشكيل format البيانات فى شكل مقبول ثم يتم إرساله إلى العميل ، كما يمكن أن تتحكم الـ middle tier فى من له الحق فى رؤية البيانات وهذه الطبقة تكون غالباً برامج web server مثل الـ IIS.
- طبقة العميل client tier وهى التى تمثل واجهة التطبيق للمستخدم user interface وغالباً ما تكون المتصفح web browser.

## الصفحات الساكنة والديناميكية Static and Dynamic web Pages:

غالبا ما تقسم صفحات الويب Web Pages إلى نوعين هما:

- صفحات الويب الساكنة Static web Pages ، وصفحات الويب الساكنة معناها الصفحات التى يكون محتواها ثابتاً ولا يتغير ولا يتوقف على أى عامل وأشهر مثال على هذا النوع هو صفحات الـ HTML (أى التى يكون إمتدادها .html أو .htm وتكون مكتوبة بالكامل بلغة HTML).

- صفحات ويب ديناميكية Dynamic web Pages ، وهذه الصفحات يتوقف محتواها على عوامل مختلفة مثل التوقيت أو المكان الجغرافي أو عوامل يحددها المستخدم وأشهر مثال على ذلك هو موقع www.msn.com فهذا الموقع مثلاً لو قمت بتحديد مكانك (من خلال الإعدادات الإقليمية Regional Settings) على أنك فى أمريكا ، فسيتم عرض صفحات الموقع باللغة الإنجليزية أما إذا كانت إعداداتك بأنك فى مصر ، فهنا تكون صفحات الويب Web Pages باللغة العربية وليست اللغة فقط هى التى تتغير ولكن أيضا الأخبار والمواضيع ، ولكى تقوم بكتابة صفحات ويب ديناميكية Dynamic Web Pages ، فيجب أن تستخدم لغة مساعدة بجانب لغة HTML وهذه اللغات تنقسم إلى فئتين هما:

1- <u>لغة تعمل جهة العميل Client side Scripting language</u> مثل لغة JavaScript ولغة VBScript ، وهذه اللغات يقوم بترجمتها وتنفيذ تعليماتها المتصفح Browser الموجود فى جهاز العميل ، وتكتب تعليمات هذه اللغة داخل صفحات الـ HTML أو كملفات مصاحبة لها.

2- <u>لغة تعمل جهة الخادم Server Side Language</u> مثل لغة ASP.net أو JSP أو PHP ، وتعليمات هذه اللغة تكون فى ملفات موجودة على الخادم web server فيقوم الخادم بترجمة وتنفيذ هذه التعليمات وتحويلها إلى تعليمات HTML ثم يقوم بإرسالها إلى المتصفح Browser الذى يقوم بفهمها وعرضها (المتصفح لا يفهم إلا لغة HTML).

وكمثال على معنى صفحات الويب الساكنة والديناميكية تابع معى المثال التالى.

<u>ملحوظة هامة:</u>

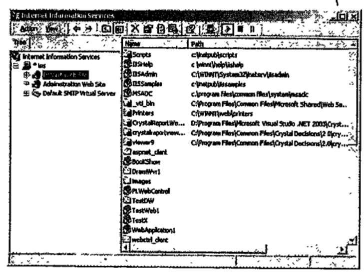
قبل البدء فى المثال يجب أن يكون خادم الويب Web Server المعروف باسم IIS مهيئاً للعمل على جهازك وللتأكد من ذلك اتبع الخطوات التالية:

## التأكد من عمل الويب سيرفر IIS

قم بالذهاب إلى نافذة التحكم في الـ IIS (نظام ويندوز 2000 فما أعلى) وذلك عن طريق

Start→programs→Administrative tools→ Internet Services Manager

فقم بفتح محتويات الخادم Server من الجزء اليساري Tree حتى تصل إلى Default web server فقم بالضغط عليه ثم قم بالذهاب إلى الزر المحاط بمربع (ويأخذ شكل مثلث رأسه جهة اليمين) كما هو واضح في الشكل 2 ، فإذا كان غير نشط فمعنى هذا أن الخادم Server يعمل أما إذا كان نشطاً فقم بالضغط عليه لتشغيل خادم IIS.



شكل 2 خادم IIS

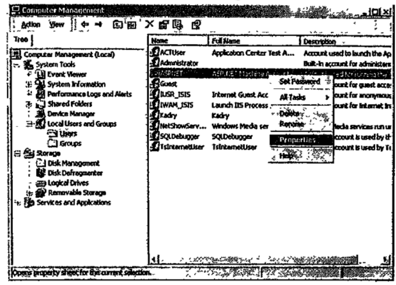
## التأكد من عمل الـ ASP.net مع قواعد البيانات:

- قم بالذهاب إلى

Start→programs→Administrative tools→ Computer Management

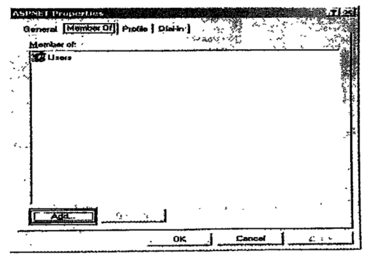
- فتجد نافذة التحكم قد فتحت فقم بالذهاب إلى الجزء اليساري Tree ثم Local Users and Groups ثم اضغط مرتين Double-click على

المجلد users ثم قم بالذهاب الى الجزء الأيمن ثم اضغط بالزر الأيمن للفارة Mouse على المستخدم ASP.NET ثم اضغط Properties من القائمـــة المختصرة وذلك كما هو واضح فى الشكل 3.



شكل 3 عمل ASP .NET مع قواعد البيانات

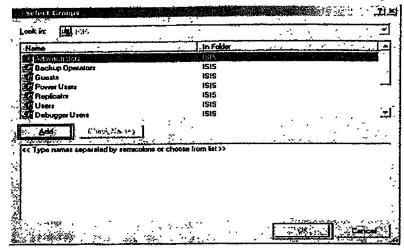
فتجد أن نافذة خواص المستخدم ASP.NET Properties قد ظهرت فقـــم بالضغط على التبويب Member Of من أعلى النافذة وذلك كما هو واضح فى الشكل 4.



شكل 4 عمل ASP .NET مع قواعد البيانات

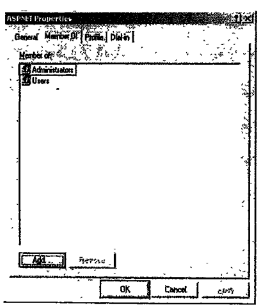
ثم قم بالضغط على الزر Add أسفل الشاشة.

- فتجد أن نافذة الـ Select Groups قد ظهرت فقم بالضغط على العنصر Administrators ثم قم بالضغط على الزر ADD ثم الزر OK وذلك كما هو واضح في الشكل 5.




شكل 5 عمل ASP .NET مع قواعد البيانات

- فتجد أن العنصر Administrators قد أضيف وذلك كما هو واضح في الشكل 6، ثم اضغط الزر OK أسفل النافذة لترجع إلى النافذة Computer Management.



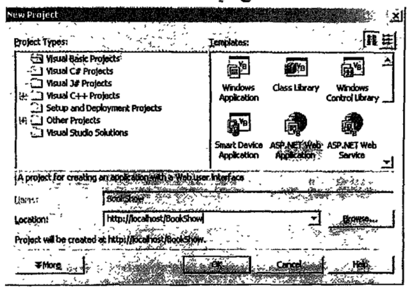

شكل 6 عمل ASP .NET مع قواعد البيانات

- بعد ذلك قم بإغلاق النافذة Computer Management ثم قم بإعادة تشغيل جهازك Restart.

## مثال 1: تطبيق ASP .NET:

قم بتشغيل بيئة Visual Studio.net وذلك عن طريق
Start→Programs→ Microsoft Visual Studio .NET → Microsoft Visual Studio .NET

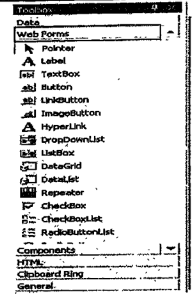
قم بإنشاء مشروع جديد من نوع ASP.net Web Application وقم بتسميته BookShow وذلك كما هو واضح في شكل 7.




شكل 7 إنشاء المشروع

فتجد أنه تم فتح شاشة بها نموذج Form يتشابه مع نماذج النوافذ Windows Forms ويسمى نموذج الويب Web Form وهى صفحة ASP.net ، وكل مشروع من هذا النوع دائماً يبدأ بصفحة من نوع ASP.net ، وهذه الصفحات ذات امتداد aspx.

ثم من على يسار الشاشة تجد صندوق الأدوات وبه العديد من الأدوات وفى هذا المثال سنقوم بالعمل مع أدوات نموذج الويب web Forms فقم بالضغط على عنوان هذه الفئة وذلك كما هو واضح فى شكل 8.




شكل 8 عمل ASP .NET مع قواعد البيانات

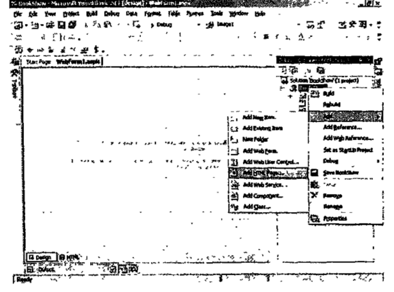
## سنقوم الآن بإنشاء صفحة ويب ساكنة Static Web Page:

- كما قلنا أن صفحات الويب الساكنة هى من نوع HTML ، ولإضافة صفحة من هذا النوع ، فقم بالذهاب إلى نافذة الـExplorer Solution ثم قم بالوقوف على اسم المشروع ثم قم بالضغط بالزر الأيمن للفارة Mouse ثم اختر

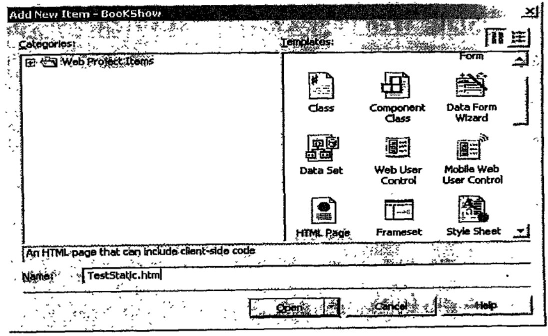
ADD→Add Html Page

وذلك كما هو واضح فى شكل 9.

- فيظهر مربع حوار Dialog Box يطلب منك اسم الملف فقم بتسميته TestStatic.htm (لاحظ الامتداد لأنه فى غاية الأهمية) وذلك كما هو واضح فى شكل 10.
- بعد ذلك اضغط الزر OK فتجد أنه تمت إضافة صفحة جديدة إلى نافذة مستكشف الحلول Solution Explorer.

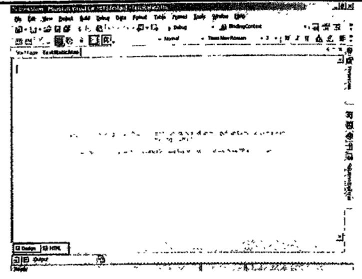


شكل 9 إضافة صفحة HTML




شكل 10 إضافة صفحة HTML

- فتجد أنه تم إنشاء صفحة بيضاء عنوانها TestStatic.htm وذلك كما هو واضح في شكل 11.

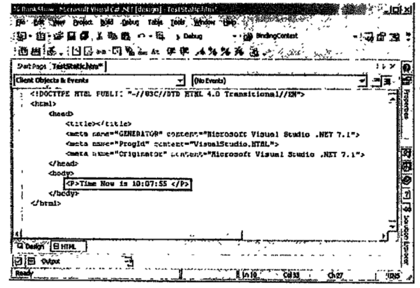


شكل 11 إضافة صفحة HTML

- قم بضغط الوصلة HTML (المحاطة بالمستطيل الأحمر فى الشكل) فتجد أن نافذة كود ظهرت ، وهذا الكود هو كود الـ HTML المكون للصفحة ، فقم بكتابة السطر التالى بعد العلامة <BODY>

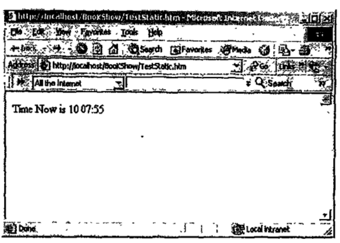
<P>Time Now Is 10:07:55</P>

وذلك كما هو واضح فى الشكل 12 ثم قم بتنفيذ المشروع (اضغط الزر F5) فماذا ترى؟




شكل 12 إضافة صفحة HTML

- ستجد أن الصفحة التي تم عرضها هي صفحة الـ ASP وليست صفحة الـ HTML التي قمنا بإنشائها ، ولتعديل ذلك قم بإغلاق المتصفح Browser ثم ارجع إلى شاشة NET. Visual Studio ثم قم بالذهاب إلى نافذة مستكشف الحل Solution Explorer ثم قف على الصفحة TestStatic.html ثم اضغط بالزر الأيمن للفارة Mouse ثم قم باختيار الأمر Set As Start Page وذلك حتى تكون هي الصفحة التي تظهر عند تنفيذنا للمشروع.
- الآن قم بتنفيذ المشروع (اضغط F5 ) فتجد أن المتصفح Browser يظهر به صفحة TestStatic.html ويعرض التوقيت كما هو واضح في شكل 13.
- انتظر قليلاً (خمس أو عشر ثوان) ثم وأنت واقف في نافذة المتصفح Browser (نافذة المتصفح Browser نشطة) ، قم بضغط الزر F5 فماذا ترى ؟



شكل 13 إضافة صفحة HTML

- كان المفروض أن ترى الوقت وقد تغير ولكن لم يحدث هذا لأن صفحات الـ HTML كما قلنا هي صفحات ساكنة أى أن محتواها لا يتغير مع تغير أى عامل مثل الوقت أو هوية المستخدم أو...

(طبعا يتغير محتواها إذا قام مصمم الصفحة بتغيير الكود).

ملحوظة:

حاول الذهاب إلى صندوق الأدوات Toolbox وحاول فتح الفئة WEB Forms وستجد أن هذه الأدوات غير نشطة لأنها تعمل فقط مع صفحات الـ ASP.net.

- قم الآن بإغلاق المتصفح Browser ثم عد إلى شاشة Visual Studio ..NET

## العمل مع صفحات الويب الديناميكية ASP.net:

- سنقوم الآن بالعمل مع صفحة الـ Asp.net فقم بالذهاب الى مستكشف الحلول Solution Explorer ثم قم بالوقوف على الصفحة WebForm1.aspx ثم اضغط بالزر الأيمن للفارة Mouse واختر Set As Start Page وذلك حتى تكون هذه الصفحة هى التى تظهر عند تنفيذ البرنامج ، ثم قم بالضغط مرتين Double-click على الصفحة وذلك حتى تكون هى النشطة فى حالة التصميم.
- قم بالذهاب إلى صندوق الأدوات Toolbox ثم من الفئة WEB Forms (تسمى هذه الأدوات أيضا بأدوات الخادم Server Controls) قم بسحب أداة العنوان Label وضعها على النموذج Form.
- قم بالضغط على الزر F4 وذلك لكى تظهر شاشة الخصائص Properties للأداة Label.
- قم بتغيير الخصائص التالية إلى:

Width= 296px
Height=48px

- قم الآن بضغط الزر F7 وذلك للذهاب لنافذة الكود.

- قم بالبحث عن الدالة ()Page_Load (وهى دالة Method تنفذ عند بداية تحميل الصفحة) فى الكود ثم اجعل الدالة Method تبدو كما فى السطور التالية:

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
Label1.Text = String.Format("{0:D2}:{1:D2}:{2:D2}",
DateTime.Now.Hour, DateTime.Now.Minute, DateTime.Now.Second)
End Sub
```

- قم الآن بتنفيذ الكود (اضغط الزر F5) فتجد أن نافذة المتصفح Browser قد فتحت وبها يظهر التوقيت.
- انتظر خمس أو عشر ثوان ثم قم بضغط الزر F5 فتجد أن التوقيت قد اختلف تبعا للوقت الذى تم ضغط الزر F5 فيه. حاول أن تكرر نفس العملية بعد فترة أخرى ستجد أن التوقيت يختلف في كل مرة.
- قم بإغلاق المتصفح Browser وعد إلى شاشة Visual Studio.

بعد أن تعرفنا بشكل بسيط على الفرق بين الصفحات الساكنة والصفحات الديناميكية تعال معى عزيزى القارئ نغوص بشكل أعمق فى صفحات وأدوات الـ ASP.net.

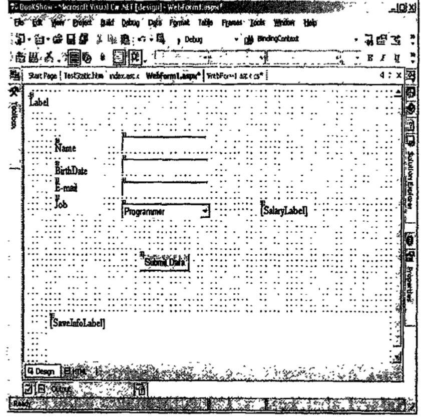
## العمل مع صفحات وأدوات الـ ASP.net:

قم بإضافة الأدوات الموجودة فى الجدول التالى مع تغيير خصائصها كما هو موجود فى الجدول وكما هو موجود فى الشكل 14.

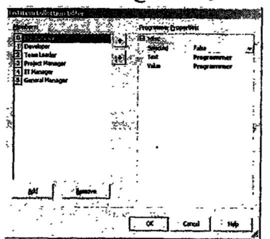
| الأداة | الخاصية | قيمة الخاصية |
|---|---|---|
| Label 1 | Text | Name |
| | Width | 91px |
| | Height | 19px |

| BirthDate | Text | Label 2 |
|---|---|---|
| 91px | Width | |
| 19px | Height | |
| E-Mail | Text | Label 3 |
| 91px | Width | |
| 19px | Height | |
| Job | Text | Label 4 |
| 91px | Width | |
| 19px | Height | |
| | Text | Label 5 |
| 100px | Width | |
| 24px | Height | |
| SalaryLabel | ID | |
| 600px | Width | Label 6 |
| 72px | Height | |
| SaveInfoLabel | ID | |
| 155px | Width | TextBox1 |
| 19px | Height | |
| NameTextBox | ID | |
| 155px | Width | TextBox2 |
| 19px | Height | |
| BirthDateTextBox | ID | |
| 155px | Width | TextBox3 |
| 19px | Hieght | |
| EMailTextBox | ID | |
| 155px | Width | DropDownList |
| 19px | Height | |
| JobCompo | ID | |
| 88px | Width | Button |
| 30px | Height | |
| Submit Data | Text | |
| SubmitDataButtton | ID | |

الغرض من هذا النموذج Form هو استقبال بيانات من المستخدم ثم تخزينها.
بجانب خصائص الأدوات التى فى الجدول السابق نحتاج أن ندخل عناصر القائمة DropDownList وذلك كالأتى:
قم بالضغط على الأداة DropDownList الموجودة على النموذج Form ثم قم بضغط الزر F4 وذلك لإظهار شاشة الخصائص Properties Window.
قم بالذهاب إلى الخاصية Items ثم قم بضغط الزر المنقط فتجد أنه تم ظهور مربع حوار Dialog Box كما هو واضح فى شكل 15 ، ثم قم بضغط الزر ADD وذلك لإضافة عنصر item فى الأداة ، ثم فى الجزء الأيمن قم بكتابة اسم العنصر Programmer فى الخاصية Text ثم اضغط الزر Add مرة أخرى لإضافة عنصر أخر (قم بإدخال العناصر الموجودة فى الشكل 15) وبعد أن تنتهى من إدخال كافة العناصر قم بضغط الزر OK.




شكل 14 إنشاء نموذج الويب Web Form




شكل 15 إضافة عناصر القائمة DropDownList

والغرض من هذه القائمة DropDownList أنه عند اختيار أى عنصر من عناصرها يتم كتابة المرتب الخاص به فى أداة العنوان SalaryLabel.

وللقيام بذلك قم بالضغط مرتين Double-click على الأداة DropDownList (JobCompo) فتجد أنك انتقلت لنافذة كتابة الكود ، فقم بجعل الدالـــة Method تبدو كما هو واضح فى السطور التالية:

```
Private Sub JobCompo_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles JobCompo.SelectedIndexChanged
1.     Select Case (JobCompo.SelectedIndex)
2.     Case 0
3.     SalaryLabel.Text = "Prgrammer Salary=1500"
4.     Case 1
5.     SalaryLabel.Text = "Developer Salary=2500"
6.     Case 2
7.     SalaryLabel.Text = "TeamLeader Salary=5000"
8.     Case 3
9.     SalaryLabel.Text = "ProjectManager Salary=10000"
10.    Case 4
11.    SalaryLabel.Text = "IT Manager Salary=15000"
12.    Case 5
13.    SalaryLabel.Text = "General Manager Salary=20000"
14.    End Select
End Sub
```

**شرح الكود:**

- كما ترى عزيزى القارئ ، فإننا قمنا باستخدام البلوك select...case وذلك لاختبار القيمة التى يختارها المستخدم ، والجملة select تقوم باختبار ترتيب العناصر الموجودة فى القائمة DropDownList وذلك عن طريق الجملـــة الموجودة فى السطر 1 حيث يتم فحص ترتيب العنصر عن طريق الخاصية SelectedIndex مع ملاحظة أن ترتيب أول عنصر دائماً يكون صفراً.

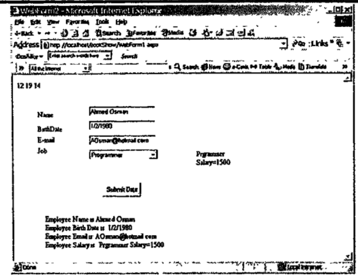
- فى السطر 2 يتم اختبار ما إذا كان المستخدم قد قام باختيار العنصر الأول الذي ترتيبه 0 (أي القيمة Programmer) ، فإذا تحقق ذلك يقوم بجعل قيمة الخاصية Text للأداة SalaryLabel تساوى Prgrammer Salary=1500 وذلك كما هو واضح فى السطر 3 .
- فى السطر 4 يتم اختبار ما إذا كان المستخدم قد قام باختيار العنصر الثاني وقيمة ترتيبه 1 (أي القيمة Developer) ، فإذا تحقق ذلك يقوم بجعل قيمة الخاصية Text للأداة SalaryLabel تساوى Developer Salary=2500 وذلك كما هو واضح فى السطر 5.
- وهكذا دواليك فى باقى الاختيارات.
- قم الآن بتنفيذ المشروع وحاول أن تقوم بتغيير الاختيارات فى القائمة المنسدلة DropDownList فى المتصفح Browser وانظر ماذا ترى؟
- أراك مندهشاً عزيزى القارئ لأنك قمت بتغيير الاختيارات مرة تلو الأخرى ولم يتم عرض الراتب فى الأداة SalaryLabel أى أنه لم يتم تنفيذ الكود!!
- والحقيقة أنك عزيزى القارئ يجب أن تنتبه إلى أن هذه الأدوات تتم معالجتها فى الخادم Server وأنت تقوم بتغيير الاختيارات فى جهة العميل Client-Side فقط ولذلك لم يتم تنفيذ الكود ، ولجعل الخادم Server يحس بأى تغيير يجرى ، فلابد أن نستخدم خاصية تسمى AutoPostBack ، وهذه الخاصية تعمل على إعادة النموذج Form إلى الخادم Server ليقوم بمقارنة أى تغيير فى حالة الأدوات.
- لذلك قم بإغلاق المتصفح Browser وعد إلى شاشة Visual Studio ثم قم باختيار Select الأداة JobCompo ثم قم بضغط الزر F4 وذلك لإظهار شاشة الخصائص Properties Window ثم قم بالذهاب إلى الخاصية AutoPostBack وقم بتغيير قيمتها إلى True.

- قم الآن بإعادة تنفيذ الكود RUN ثم قم بـــالتغيير فـــى القائمـــة المنســدلة JobCompo وانظر ماذا ترى؟
- والخاصية AutoPostBack متوفرة لجميع الأدوات تقريباً.
- والآن لنكتب الكود الذى يؤديه الزر Submit Data لذلك قم بالضغط مرتين Double-click عليه ثم اكتب الكود التالى:

```
Private Sub SubmitDataButtton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SubmitDataButtton.Click
SaveInfoLabel.Text="Employee Name is "+
NameTextBox.Text+"<BR> Employee Birth Date is: "+
BirthDateTextBox.Text+"<BR> Employee Email is: "+
EMailTextBox.Text+
"<BR> Employee Salary is : "+SalaryLabel.Text
End Sub
```

كما هو واضح من الكود أنه عند الضغط على الزر Submit Data ، فإنه يـــتم إظهار البيانات فى أداة العنوان SaveInfoLabel ، ولكن أود ان ألفت انتباهــك عزيزى القارئ للعلامة <BR> وهى علامة tag من علامـــات لغـــة HTML تؤدى إلى كتابة النص فى سطر جديد.

قم الآن بتنفيذ البرنامج وحاول إدخال بيانات ثم قم بالضغط على الزر Submit Data وستجد أن أداة العنوان تقوم بعرض ما أدخلته من بيانات وذلك كما هـــو واضح فى الشكل 16.




شكل 16 تنفيذ التطبيق

وبذلك نكون قد انتهينا من استخدام بعض أدوات الـــ ASP.net.

## تطبيق عملى:

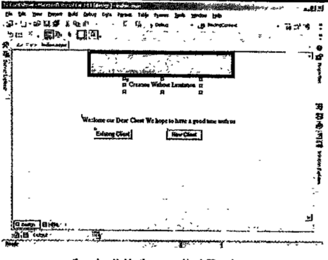
هذا التطبيق البسيط مهمته عرض بيانات موجودة فى قاعــدة بيانـــات (عـــرض أسماء كتب) ، ويتم هذا العرض للمستخدمين الذين لهم كلمة ســـر فـــى الموقـــع (مسجلين لدى الموقع).

والموقع يتكون من حمس صفحات Asp.net هم:

1. index.aspx
2. Registration.aspx
3. Security.aspx
4. Thnx.aspx
5. BookTitles.aspx

حيث:

Index.aspx هى الصفحة الافتتاحية (أول صفحة يتم عرضها) وفيها يتم سؤال المستخدم ما إذا كان مستخدماً جديداً أم مستخدم موجود وشكلها كالآتى:



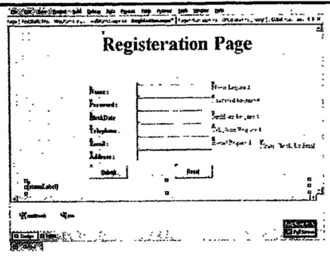
شكل 17 الصفحة الافتتاحية

فإذا كان المستخدم جديداً فيتم نقله إلى صفحة تسجيل البيانات Registeration.aspx والتي يتم فيها إدخال البيانات وشكلها كما هو واضح في شكل 18.

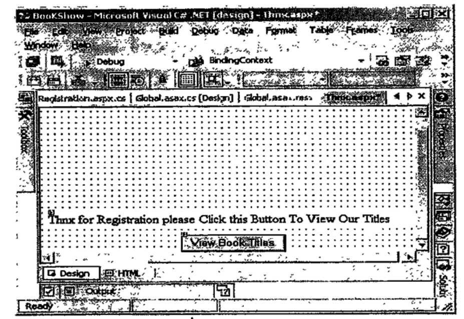
وبعد أن يتم إدخال البيانات بشكل صحيح فإنة يتم تحويلة إلى صفحة الشكر Thnx.aspx كما هو واضح في شكل 19.

ثم يقوم بضغط الزر View Book Title ليتم تحويله إلى صفحة رؤية البيانات BookTitles.aspx التى تستعيد البيانات من قاعدة البيانات التى سننشئها وذلك كما هو واضح فى شكل 20.

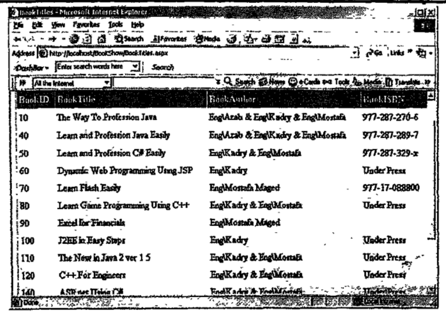
أما إذا كان المستخدم موجود بالفعل فإنة يتم تحويله لصفحة الأمن Security.aspx كما هو واضح في شكل 21 والتي يتم التأكد من بيانات المستخدم عن طريق مقارنة ما أدخله المستخدم من بيانات مع البيانات الموجودة فى قاعدة البيانات ، فإذا كان ما أدخله صحيحاً ، فيتم نقله إلى صفحة رؤية البيانات BookTitles.aspx.




شكل 18 صفحة التسجيل

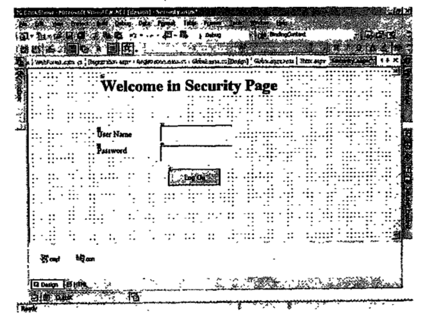



شكل 19 صفحة الشكر



| BookID | BookTitle | BookAuthor | BookISBN |
|---|---|---|---|
| 10 | The Way To Profession Java | Eng\Arab & Eng\Kadry & Eng\Mostafa | 977-287-270-6 |
| 40 | Learn and Profession Java Easily | Eng\Arab & Eng\Kadry & Eng\Mostafa | 977-287-289-7 |
| 50 | Learn and Profession C# Easily | Eng\Kadry & Eng\Mostafa | 977-287-329-x |
| 60 | Dynamic Web Programming Using JSP | Eng\Kadry | Under Press |
| 70 | Learn Flash Easily | Eng\Mostafa Maged | 977-17-088800 |
| 80 | Learn Game Programming Using C++ | Eng\Kadry & Eng\Mostafa | Under Press |
| 90 | Excel for Financials | Eng\Mostafa Maged | |
| 100 | J2EE in Easy Steps | Eng\Kadry | Under Press |
| 110 | The New in Java 2 ver 1 5 | Eng\Kadry & Eng\Mostafa | Under Press |
| 120 | C++ For Engineers | Eng\Kadry & Eng\Mostafa | Under Press |
| 140 | ASP.net Using C# | [illegible] | [illegible] |

شكل 20 صفحة البيانات




شكل 21 صفحة الأمن

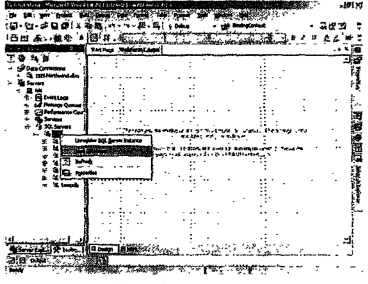
## خطوات العمل:

1. إنشاء قاعدة البيانات.
2. إنشاء الصفحات.

## إنشاء قاعدة البيانات:

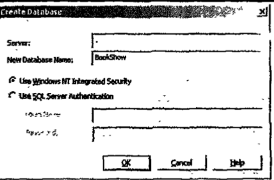
فى الخطوات القادمة سوف نقوم بإنشاء قاعدة بيانات تسمى BookShow وهى تتكون من جدولين هما Books و Registeration فاتبع الخطوات التالية:

- قم بالذهاب إلى نافذة الـ Server Explorer وقم بالوصول إلى اسم الخادم Server الخاص بك وذلك كما هو واضح فى شكل 22.
- اضغط بالزر الأيمن للفارة Mouse ثم اختر New DataBase.

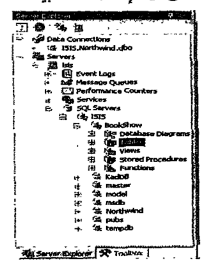


شكل 22 إنشاء قاعدة بيانات

- فيظهر لك مربع حوار Dialog Box طالباً منك اسم لقاعدة البيانات فقم بإدخال BookShow واجعل باقى الاختيارات كما هو واضح فى شكل 23 ثم إضغط الزر OK.
- ستجد أن برنامج إدارة قواعد البيانات SQL Server قد أنشأ قاعدة بيانات جديدة فقم بالضغط على علامة + بجانب اسم القاعدة لترى مكونات قاعدة البيانات وذلك كما هو واضح فى شكل 24.

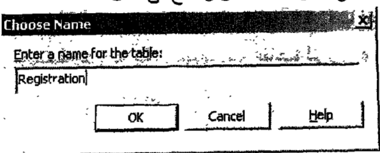



شكل 23 إنشاء قاعدة بيانات




شكل 24 إنشاء قاعدة بيانات

- اضغط بالزر الأيمن للفارة Mouse على المجلد Tables ثم اختـر New Table وذلك لإنشاء جدول جديد في قاعدة البيانات.
- يظهر لك مربع حوار Dialog Box طالباً منك اسم الجدول.فقـم بتسـميته Registeration وذلك كما هو واضح في شكل 25.



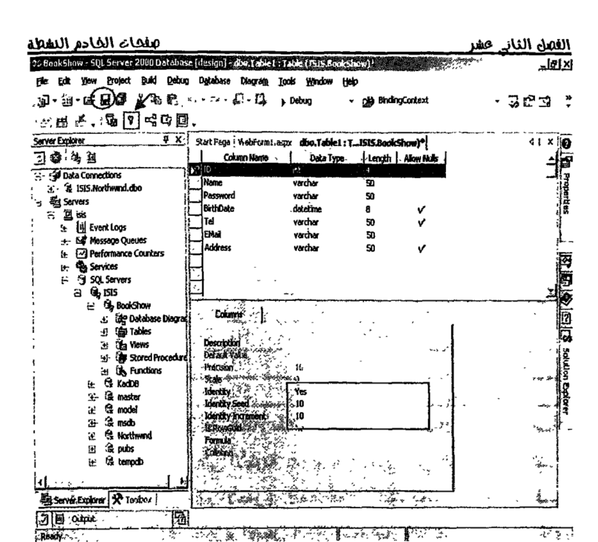

شكل 25 إنشاء قاعدة بيانات

■ ستجد أن الجانب الأيمن به نافذة تصميم الجدول فقم بإدخال أسماء الحقول وأنواعها كما هو واضح فى الجدول التالى وفى شكل 26.

| Column name | Data Type | Length | Allow Null |
|---|---|---|---|
| ID | int | 4 | No |
| Name | varchar | 50 | No |
| Password | Varchar | 50 | No |
| BirthDate | DateTime | 8 | Yes |
| Tel | Varchar | 50 | Yes |
| Email | Varchar | 50 | No |
| Address | Varchar | 50 | yes |

لاحظ أن الحقل الأول ID قد جعلناه مفتاحاً أساسياً Primary Key وذلك بأن اخترنا الحقل ثم ضغطنا على الأيقونة icon التى تحمل رمز المفتاح أعلى النافذة (الزر الذى يشير إليه السهم الأحمر). لاحظ أيضاً أنك يجب أن تغير فى خصائص هذا الحقل فى الجزء الأسفل (اجعل الخواص كما تبدو فى الجزء الذى حوله مستطيل أحمر فى الشكل 26).

■ قم بعد ذلك بحفظ التصميم وذلك عن طريق الزر المحاط بدائرة حمراء كما هو واضح فى الشكل 26.



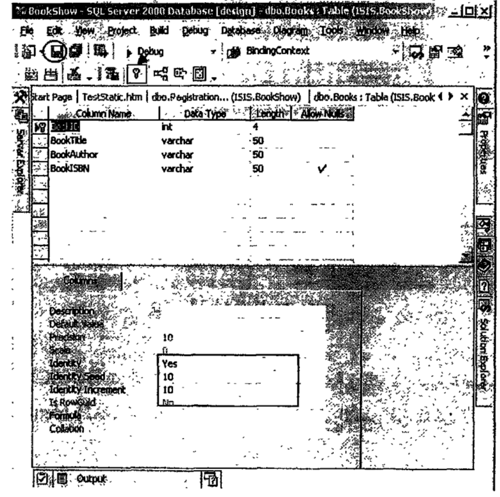
شكل 26 إنشاء قاعدة بيانات

- بهذا نكون قد أنشأنا الجدول الأول.
- قم بنفس الخطوات السابقة لإنشاء الجدول Books وحقوله كما هو واضح فى الجدول التالى وفى شكل 27.

| Column name | Data Type | Length | Allow Null |
|---|---|---|---|
| BookID | int | 4 | No |
| BookTitle | varchar | 50 | No |
| BookAuthor | Varchar | 50 | No |
| BookIsbn | VarChar | 50 | Yes |

لاحظ أن الحقل BookID له خاصية المفتاح الأساسي Primary Key ، فقم بنفس العمل له كما فعلنا مع الحقل id في الجدول السابق (ولا تنس إعداد الخواص المحاطة بالمستطيل الأحمر أسفل الشكل).

بعد تصميم الجدول قم بحفظ الإعدادات وذلك عن طريق الزر المحاط بدائرة حمراء.

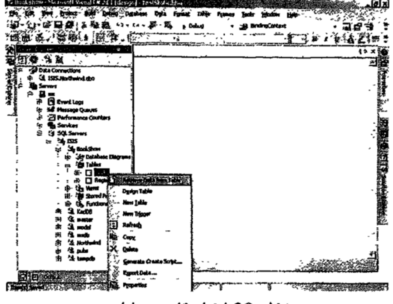



شكل 27 إنشاء قاعدة بيانات

- بعد ذلك نود إدخال بيانات في الجدول BOOKS ويتم ذلك بأن نذهب إلى نافذة الـ Server Explorer ثم نقف على الجدول BOOKS ثم نضغط

بالزر الأيمن للفارة Mouse ثـم نختار Retrieve Data From Table وذلك كما هو واضح فى شكل 28.

- ستجد أن الجدول أصبح فى حالة إدخال البيانات فقم بإدخال البيانات الموضحة فى الشكل 29 أو قم بإدخال بيانات من عندك بما يلائم أنواع الحقول.



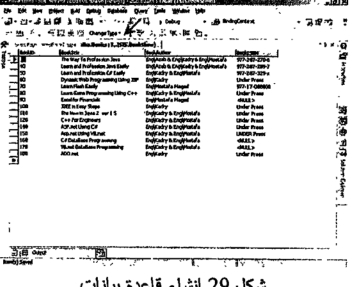
شكل 28 إنشاء قاعدة بيانات

ملحوظة:

الحقل BOOKID تتم إدخال بياناته بطريقة أوتوماتيكية وذلك لأن لـه خاصية الـ Identity (الخاصية التى كانت محاطة بالمستطيل الأحمر فى الشكل 27).

بعد إدخال البيانات قم بضغط الأيقونة icon ذات علامة التعجب الحمراء ( Run Query) أعلى النافذة (الذى يشير إليها السهم الأحمر فى الشكل 29).

بذلك نكون قد انتهينا من تصميم قاعدة البيانات وبقى الجزء الخاص بصفحات الـ ASP.net فتابع معى الفقرة القادمة.



شكل 29 إنشاء قاعدة بيانات

## العمل مع صفحات الـ ASP.net:

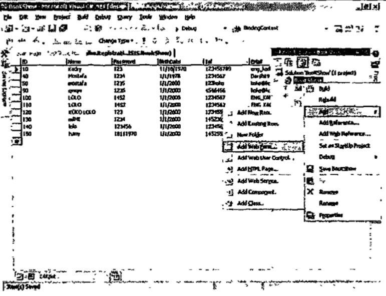
سنقوم الآن بإضافة خمس صفحات ASP.net للمشروع الحالي:

- قم بالذهاب إلي نافذة الـ Solution Explorer ثم قم بالوقوف على اسم المشروع ثم قم بالضغط بالزر الأيمن للفارة Mouse ثم اختر
ADD→Add Web Form

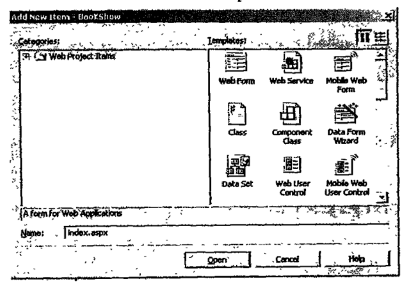
وذلك كما هو واضح في شكل 30.

- فيظهر لك مربع حوار Dialog Box طالباً اسم النموذج Form فاكتب index.aspx وذلك كما هو واضح في شكل 31.
- كرر العملية السابقة لإضافة الصفحات

- Registration.aspx
- Security.aspx
- Thnax.aspx
- BookTitles.aspx



شكل 30 إنشاء صفحات ASP




شكل 31 إنشاء صفحات ASP

- بعد ذلك قم بالذهاب إلى نافذة الـ Solution Explorer ثم قم بالضغط بالزر الأيمن للفارة Mouse على الصفحة index.aspx ثم اختر Set as Start Page وذلك لجعلها أول صفحة تعمل عند تنفيذ التطبيق ، ثم قم بعد ذلك بالضغط مرتين Double-click عليها وذلك لجعلها تظهر في حالة التصميم Design Mode.

## العمل مع الصفحة index.aspx:

قم بإضافة الأدوات وقم بتغيير خصائصها كما هو موجود فى الجدول التالى لتبدو كما هو واضح فى الشكل 17 السابق.

| قيمة الخاصية | الخاصية | الأداة |
|---|---|---|
| Welcome To KM Systems | Text | Label 1 |
| px369 | Width | |
| px64 | Height | |
| X-Large | Font | |
| (web)Silver | BackColor | |
| (web)Black | Border Color | |
| Creation Without Limitation | Text | Label 2 |
| px184 | Width | |
| px24 | Height | |
| Weclome our Dear Client We hope to Spend a good time with us | Text | Label 3 |
| px391 | Width | |
| px24 | Height | |
| ExistingClientButton | ID | Button |
| px96 | Width | |
| 30px | Height | |
| Existing Client | Text | |
| NewClientButton | ID | Button |
| 96px | Width | |
| 30 | Height | |
| New Client | Text | |

وسنقوم الآن بكتابة الكود الخاص بالزر Existing Client ومن المفترض أنـــه عند الضغط علي هذا الزر أن يتم الانتقال إلى الصفحة Security.aspx ، لذلك

قم بالضغط مرتين Double-click عليه بالفارة Mouse ، فقم بجعل دالة الحدث ExistingClientButton_Click() تبدو كما في الكود التالي:

```
Private Sub ExistingClientButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ExistingClientButton.Click
    Server.Transfer("Security.aspx")
End Sub
```

كما ترى في الكود أننا استخدمنا الهدف Server وهو هدف object مدمج في الـ ASP (جاهز للعمل بدون إنشاء) ولهذا الهدف object دالة Method هي Transfer() وتستخدم في التنقل بين الصفحات المختلفة للتطبيق ، وهذه الدالة Method تستقبل معاملاً parameter يمثل اسم الصفحة المراد الانتقال إليها.

بالمثل قم بالضغط مرتين Double-click على الزر New Client ، ومن المفترض أنه عند الضغط علي هذا الزر أن يتم الانتقال إلى الصفحة Registration.aspx.

```
Private Sub NewClientButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles NewClientButton.Click
Server.Transfer("Registration.aspx")
End Sub
```

والكود الكامل للصفحة هو

```
Public Class Index
   Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

   'This call is required by the Web Form Designer.
   <System.Diagnostics.DebuggerStepThrough()> Private
```

```
Sub InitializeComponent()

   End Sub
   Protected WithEvents Label1 As
System.Web.UI.WebControls.Label
   Protected WithEvents Label2 As
System.Web.UI.WebControls.Label
   Protected WithEvents Label3 As
System.Web.UI.WebControls.Label
   Protected WithEvents ExistingClientButton As
System.Web.UI.WebControls.Button
   Protected WithEvents NewClientButton As
System.Web.UI.WebControls.Button

   'NOTE: The following placeholder declaration is required
by the Web Form Designer.
   'Do not delete or move it.
   Private designerPlaceholderDeclaration As
System.Object

   Private Sub Page_Init(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Init
      'CODEGEN: This method call is required by the Web
Form Designer
      'Do not modify it using the code editor.
      InitializeComponent()
   End Sub

#End Region

   Private Sub Page_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      'Put user code to initialize the page here
```

```
End Sub

Private Sub ExistingClientButton_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ExistingClientButton.Click
    Server.Transfer("Security.aspx")
End Sub

Private Sub NewClientButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NewClientButton.Click
    Server.Transfer("Registration.aspx")
End Sub
End Class
```

حاول أن تقوم بتنفيذ RUN الصفحة وقم بالضغط على الأزرار وستجد أنك انتقلت للصفحة المرتبطة بالزر الذى ضغطت عليه (انظر في شريط العنوان Address Bar).

## العمل مع الصفحة Security.aspx:

قم بالذهاب إلى نافذة الـ Solution Explorer وقم بالضغط مرتين Double-click على الصفحة Security.aspx لتصبح فى حالة التصميم وقم بإضافة الأدوات التالية مع تغيير خصائصها كما هو واضح فى الجدول التالى ولتصبح كما هو واضح في شكل 21 السابق.

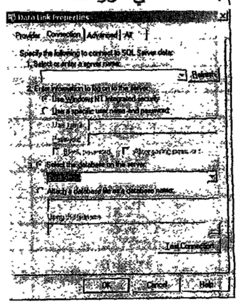
| قيمة الخاصية | الخاصية | الأداة |
|---|---|---|
| Welcome in Security Page | Text | Label 1 |
| px360 | Width | |
| px40 | Height | |

| | | |
|---|---|---|
| X-Large | Font | |
| Silver | BackColor | |
| Black | Border Color | |
| User Name | Text | Label 2 |
| px112 | Width | |
| px24 | Height | |
| Password | Text | Label 3 |
| px112 | Width | |
| px24 | Height | |
| statusLabel | ID | Label 4 |
| px328 | Width | |
| 24px | Height | |
| px104 | Width | Button |
| 30px | Height | |
| Log On | Text | |
| nameTextBox | ID | TextBox1 |
| px144 | Width | |
| 24 px | Height | |
| | Text | |
| passwordText | ID | TextBox 2 |
| px144 | Width | |
| 24 px | Height | |
| password | TextMode | |

## العمل مع أدوات قواعد البيانات:

- فى هذا النموذج Form سوف نتعامل مع قواعد البيانات ولذلك لابد من استخدام أدوات قواعد البيانات الموجودة فى الفئة Data الموجودة فى صندوق الأدوات ToolBox ، لذلك قم بسحب الأداة , SqlConnection SqlCommand وضعهما على النموذج Form وستجدهما ممثلين بأيقونتين icons أسفل النموذج Form.

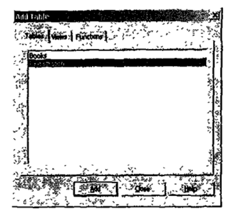
- قم باختيار الأداة SqlConnection ثم قم بضغط الزر F4 وذلــك لضــبط خصائصها وذلك كالآتي:
- قم بتغيير خاصية الاسم إلى con.
- قم بالذهاب إلي الخاصية Connection String ثــم اختــر مــن القائمــة المنسدلة new connection فيظهر مربع حوار Dialog Box كمــا هــو واضح فى الشكل 32 ، ثم قم بتغيير الإعدادات كما هو واضح فى الشــكل 32 ثم قم بالضغط على الزر Test Connection لتختبر نجاح الاتصال مع قاعدة البيانات ثم قم بالضغط علي الزر OK.



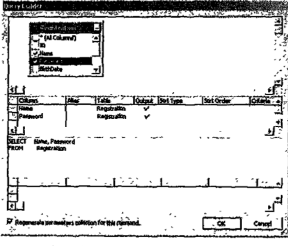
شكل 32 الاتصال بقاعدة البيانات

- قم بعد ذلك باختيار الأداة SqlCommand1 الموجــودة أســفل النمــوذج Form ثم قم بضغط الزر F4 لضبط خصائصها.
- قم بتغيير خاصية الــ Name إلى cmd.
- قم بالذهاب إلي خاصية connection ثم قم بالضغط على القائمة المنسدلة ثم قم بالضغط على علامة + الموجودة بجانب existing ثم قم باختيــار con وهى الوصلة التى أعددناها فى الخطوة السابقة.

- قم بالذهاب إلي خاصية text commnd (الأمر الذى سينفذه الأمر Sqlcommand con فى قاعدة البيانات) ثم اضغط على الزر المنقط وستجد أن هناك مربع حوار Dialog Box تم فتحه وذلك لاختيار الجدول الذى نرغب في العمل فيه فاختار Registeration ثم اضغط الزر add كما هو واضح فى الشكل 33.
- بعد ذلك نختار الحقول التي نود التعامل معها ، وبما أننا نريد اختبار الاسم وكلمة السر ، فإننا نختار الحقل name , password (عن طريق الضغط على المربع المجاور لاسم الحقل فى الجدول "علامة الصح") كما هو واضح فى شكل 34.



شكل 33 الاتصال بقاعدة البيانات



شكل 34 الاتصال بقاعدة البيانات

بعد اختيار الحقول قم بضغط الزر Ok.

بذلك نكون قد انتهينا من عمل الأدوات التي سنستخدمها في العمل مع قواعد البيانات.

والكود التالي هو كود النموذج Form وهو يماثل ما هو موجود لديك فيما عدا المظلل منه فقم بإضافته وهو ماسنقوم بشرحه.

```
1.   Imports System.Data.SqlClient
2.   Public Class Security
3.   Inherits System.Web.UI.Page
4.   Dim name As String
5.   Dim password As String
6.   Dim dr As SqlDataReader
7.   #Region " Web Form Designer Generated Code "

8.   'This call is required by the Web Form Designer.
9.   <System.Diagnostics.DebuggerStepThrough()>
Private Sub InitializeComponent()
10.  Me.cmd = New System.Data.SqlClient.SqlCommand
11.  Me.con = New System.Data.SqlClient.SqlConnection
'
'cmd
'
12.  Me.cmd.CommandText = "SELECT Name, Password
FROM Registeration"
13.  Me.cmd.Connection = Me.con
'
'con
'
14.  Me.con.ConnectionString = "workstation
id=ISIS;packet size=4096;integrated security=SSPI;data
source=isis;pe" & _
```

```
"rsist security info=False;initial catalog=BookShow"

15. End Sub
16. Protected WithEvents Label1 As
System.Web.UI.WebControls.Label
17. Protected WithEvents Label2 As
System.Web.UI.WebControls.Label
18. Protected WithEvents Label3 As
System.Web.UI.WebControls.Label
19. Protected WithEvents StatusLabel As
System.Web.UI.WebControls.Label
20. Protected WithEvents LogonButton As
System.Web.UI.WebControls.Button
21. Protected WithEvents nameTextBox As
System.Web.UI.WebControls.TextBox
22. Protected WithEvents passwordText As
System.Web.UI.WebControls.TextBox
23. Protected WithEvents cmd As
System.Data.SqlClient.SqlCommand
24. Protected WithEvents con As
System.Data.SqlClient.SqlConnection

25. 'NOTE: The following placeholder declaration is
required by the Web Form Designer.
26. 'Do not delete or move it.
27. Private designerPlaceholderDeclaration As
System.Object

28. Private Sub Page_Init(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Init
```

```
29. 'CODEGEN: This method call is required by the Web
Form Designer
30. 'Do not modify it using the code editor.
31. InitializeComponent()
32. End Sub

33. #End Region

34. Private Sub Page_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
35. name = nameTextBox.Text.ToUpper()
36. password = passwordText.Text.ToUpper()
37. End Sub

38. Private Sub LogonButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
LogonButton.Click
39. Dim str1 As String
40. Dim str2 As String
41. Try
42. con.Open()
43. dr = cmd.ExecuteReader
44. While (dr.Read)
45. str1 = CType(dr(0), String)
46. str1 = str1.ToUpper
47. str2 = CType(dr(1), String)
48. str2 = str2.ToUpper
49. If (str1.Equals(name) And str2.Equals(password))
Then Response.Redirect("BookTitles.aspx")
50. Else
```

```
51. StatusLabel.Text = ("<I>Wrong User Name or
password</I>")
52. End If
53. End While
54. Catch ex As Exception
55. StatusLabel.Text = "Error in DataBase :" +
ex.Message
56. Finally
57. dr.Close()
58. con.Close()
59. End Try
60. End Sub
61. End Class
```

**شرح الكود:**

- فى السطر 1 قمنا باستيراد المكتبة

imports System.Data.SqlClient;

وذلك لأننا سنستخدم إحدى الفصائل الموجودة فيها وهى الفصيلة SqlDataReader.

- فى السطر 4 نجد الجملة

Dim name As String

قمنا بالإعلان عن المتغير name من نوع string وذلك لتخزين الإسم الذى سيدخله المستخدم فى مربع النص (nameTextBox).

- فى السطر 5 نجد الجملة

Dim password As String

- وفيها قمنا بالإعلان عن المتغير password وذلك لتخزين كلمة السر التى سيدخلها المستخدم.

- فى السطر 18 نجد الجملة

Dim dr As SqlDataReader

قمنا بالإعلان عن مؤشر dr من نوع الفصــيلة SqlDataReader وهــى الفصيلة التي يتم فيها تخزين البيانات التى تنتج عن تنفيذ الأمر cmd.

- فى السطر 35 نجد الجملة

name=nameTextBox.Text.ToUpper();

نقوم بتخزين القيمة التى سيدخلها المستخدم فى الأداة nameTextBox بعد تحويل حروفها إلى حروف كبيرة Capital باستخدام الدالــة ToUpper() بتخزينها فى المتغير name .

- فى السطر 36 نجد الجملة

password=passwordText.Text.ToUpper();

نقوم بتخزين القيمة التى سيدخلها المستخدم فى الأداة passwordText بعد تحويل حروفها الى حروف كبيرة Capital باستخدام الدالــة ToUpper() بتخزينها فى المتغير password ، وقد كتبنا السطرين 35 و36 داخل الدالة Page_Load() ليتم تنفيذهما عند بداية تحميل النموذج Form Load.

- فى السطور من 39 إلى 59 يتم كتابة أوامر عمل الــزر Log on وهــى كالآتى.

- فى السطرين 39 و40 أنشأنا متغيــرين مــن نــوع String وذلــك لأننــا سنســتخدمهما فــى تخــزين بيانــات اســم المســتخدم وكلمــة الســر (username,password) الموجودة فى الهدف da (الهدف الذى يحتــوى البيانات الناتجة عن تنفيذ الأوامر الموجودة فى الهدف cmd).

- فى السطر 42 نقوم بفتح الوصلة مع قاعدة البيانات باستخدام الهــدف con عن طريق استدعاء الدالة open().

- فى السطر 43 نستخدم الهدف cmd فى تنفيذ جملة الــ SQL الذى يحملها وهــى("select name,password from Registeration ") وذلــك

باستدعاء الدالة ExecuteReader() التي تقوم بإرجاع هدف object من نوع SqlDataReader ويتم تخزينه في المؤشر dr.

- في السطور 44 إلى 53 نستخدم الدوارة While loop في قراءة السجلات Records الموجودة في الهدف dr والشرط المستخدم هو dr.Read() ومعناه طالما أن بداخل الهدف dr سجلات ، فاستمر بالقراءة.
- في السطر 45 نقوم بعمل تحويل casting للقيمة الموجودة في العمود dr[0] (الذي يشير إلى عمود الاسم name) إلى String ثم نقوم بتخزينها في المتغير str1 وذلك بأستخدام الدالة CTYPE() والتي تأخذ معاملين الأول النوع المراد تحويلة والثاني النوع المراد التحويل إلية.
- في السطر 46 نقوم بتحويل الحروف الموجودة في المتغير str1 إلى حروف كبيرة Capital.
- في السطر 47 نقوم بعمل تحويل casting للقيمة الموجودة في العمود dr[1] (الذي يشير الى عمود كلمة السر password) إلى string ثم نقوم بتخزينها في المتغير str2 كما في سطر 45.
- في السطر 49 نقوم بأختبار القيمة التي أدخلها المستخدم والمخزنة في المتغيرين name,paswrd مع القيم التي تم قراءتها من قاعدة البيانات والمخزنة في المتغيرين str1,str2 ، فإذا كانت القيم متطابقة فيتم تحويل المستخدم إلى صفحة رؤية عناوين الكتب BookTitles.aspx وذلك عن طريق الجملة

```
Response.Redirect("BookTitles.aspx");
```

أما إذا لم تكن القيم متطابقة فيتم ظهور رسالة خطأ في أداة العنوان statusLabel.

- فى السطرين 57 و58 يتم إغلاق الوصلة con والهدف dr باستخدام الدالة .Close()

وبهذا نكون قد انتهينا من هذا النموذج Form.

## العمل مع النموذج Registration.aspx:

قم بالذهاب إلى نافذة الـ Solution Explorer وقم بالضغط مرتين Double-click على الصفحة Registration.aspx لتصبح فى حالة التصميم وقم بإضافة الأدوات التالية مع تغيير خصائصها كما هو واضح فى الجدول التالى ولتصبح كما هو واضح فى شكل 18 السابق.

| قيمة الخاصية | الخاصية | الأداة |
|---|---|---|
| Registeration Page | Text | Label 1 |
| px393 | Width | |
| px56 | Height | |
| X-Large | Font | |
| Silver | BackColor | |
| Black | Border Color | |
| Name: | Text | Label 2 |
| px112 | Width | |
| 16px | Height | |
| Passowrd: | Text | Label 3 |
| 112px | Width | |
| 16px | Height | |
| BirthDate: | Text | Label 4 |
| px112 | Width | |
| 16px | Height | |
| Telephone : | Text | Label 5 |
| px112 | Width | |
| 16px | Heigth | |
| Email : | Text | Label 6 |

| px112 | Width | |
|---|---|---|
| 16px | Height | |
| Address: | Text | Label 7 |
| px112 | Width | |
| 16px | Height | |
| statusLabel | ID | Laebl 8 |
| px712 | Width | |
| px24 | Height | |
| NameTextBox | ID | TextBox 1 |
| 192 | Width | |
| px24 | Height | |
| | | |
| PasswordTextBox | ID | TextBox 2 |
| 192 | Width | |
| px24 | Height | |
| BirthDateTextBox | ID | TextBox 3 |
| 192 px | Width | |
| px24 | Height | |
| TelephoneTextBox | ID | TextBox 4 |
| 192px | Width | |
| px24 | Height | |
| EMailTextBoxl | ID | TextBox 5 |
| 192px | Width | |
| px24 | Height | |
| AddressTextBox | ID | TextBox 6 |
| 192px | Width | |
| px24 | Height | |
| SubmitButton | ID | Button 1 |
| 96px | Width | |
| 30 | Height | |
| Submit | Text | |

| ResetButton | ID | Button 2 |
|---|---|---|
| 96px | Width | |
| 30 | Height | |
| Reset | Text | |
| Name Required | Error Message | RequiredFieldValidator 1 |
| NameTextBox | Control to Validate | |
| Password Required | Error Message | RequiredFieldValidator 2 |
| PasswordTextBox | Control to Validate | |
| BirthDate Required | Error Message | RequiredFieldValidator 3 |
| BirthDateTextBox | Control to Validate | |
| Telephone Required | Error Message | RequiredFieldValidator 4 |
| TelephoneTextBox | Control to Validate | |
| E-mail Required | Error Message | RequiredFieldValidator 5 |
| NameTextBox | Control to Validate | |
| EMailTextBox | Error Message | RequiredFieldValidator 6 |
| NameTextBox | Control to | |

| | | |
|---|---|---|
| | Validate | |
| RegularExpressionValidator1 | Error Message | Please Check Ur Email |
| | Control to Validate | EMailTextBox |
| | ValidationExpression | Inetrnet E-mail Address |

فى هذا النموذج Form توجد أداتين لم نستخدمها من قبل وهما RequiredFieldValidator و RegularExpressionValidator. أما عمل الأولى فهو التأكد من أن المستخدم قد أدخل بيانات فى الحقل المرتبط بالأداة ، أما الثانية فتقوم بالتأكد من أن المستخدم قد أدخل البيانات طبقاً للشكل المراد ، فمثلاً البريد الإليكتروني E-mail له الصيغة التالية:

xxxx@xxxx.xxx

و RegularExpressionValidator تتأكد من المستخدم قد أدخل البيانات طبقاً لما هو متفق عليه فى الخاصية ValidationExpression.

## العمل مع أدوات قواعد البيانات:

- فى هذا النموذج Form سوف نتعامل مع قواعد البيانات ولذلك لابد من استخدام أدوات قواعد البيانات الموجودة فى الفئة Data الموجودة فى صندوق الأدوات ToolBox ، لذلك قم بسحب الأداتين SqlConnection, SqlCommand وضعهما على النموذج Form وستجدهما ممثلتين بأيقونتين icons أسفل النموذج Form.
- قم باختيار الأداة SqlConnection ثم قم بضغط الزر F4 وذلك لضبط خصائصها وذلك كالآتى:
- قم بتغيير خاصية الاسم إلى con.

- قم بالذهاب إلى الخاصية connection String ثم اختر من القائمة المنسدلة اسم الخادم Server المسمي BookShow.dbo.
- قم بعد ذلك باختيار الأداة SqlCommand1 الموجودة أسفل النموذج Form ثم قم بضغط الزر F4 لضبط خصائصها.
- قم بتغيير خاصيةالاسم Name إلى cmdInsert.
- قم بالذهاب إلى خاصية connection ثم قم بالضغط على القائمة المنسدلة ثم قم بالضغط على علامة + الموجودة بجانب existing ثم قم باختيار con وهى الوصلة التى أعددناها فى الخطوة السابقة.
- بذلك نكون قد انتهينا من عمل الأدوات التى سنستخدمها فى العمل مع قواعد البيانات.
- والكود التالى هو كود النموذج Form وهو يماثل ما هو موجود لديك فيما عدا المظلل منه فقم بإضافته وهو ماسنقوم بشرحه.

```
1.   Public Class Registration
2.   Inherits System.Web.UI.Page

3.   #Region " Web Form Designer Generated Code "

4.   'This call is required by the Web Form Designer.
5.   <System.Diagnostics.DebuggerStepThrough()>
Private Sub InitializeComponent()
6.   Me.cmdInsert = New
System.Data.SqlClient.SqlCommand
7.   Me.con = New System.Data.SqlClient.SqlConnection
8.   '
9.   'cmdInsert
10.  '
11.  Me.cmdInsert.Connection = Me.con
```

```
12. '
13. 'con
14. '
15. Me.con.ConnectionString = "workstation
id=ISIS;packet size=4096;integrated security=SSPI;data
source=isis;pe" & _
16. "rsist security info=False;initial catalog=BookShow"
17. End Sub
18. Protected WithEvents Label1 As
System.Web.UI.WebControls.Label
19. Protected WithEvents Label2 As
System.Web.UI.WebControls.Label
20. Protected WithEvents Label3 As
System.Web.UI.WebControls.Label
21. Protected WithEvents Label4 As
System.Web.UI.WebControls.Label
22. Protected WithEvents Label5 As
System.Web.UI.WebControls.Label
23. Protected WithEvents Label6 As
System.Web.UI.WebControls.Label
24. Protected WithEvents Label7 As
System.Web.UI.WebControls.Label
25. Protected WithEvents statusLabel As
System.Web.UI.WebControls.Label
26. Protected WithEvents NameTextBox As
System.Web.UI.WebControls.TextBox
27. Protected WithEvents PasswordTextBox As
System.Web.UI.WebControls.TextBox
28. Protected WithEvents BirthDateTextBox As
System.Web.UI.WebControls.TextBox
29. Protected WithEvents TelephoneTextBox As
System.Web.UI.WebControls.TextBox
```

```
30. Protected WithEvents AddressTextBox As
System.Web.UI.WebControls.TextBox
31. Protected WithEvents ResetButton As
System.Web.UI.WebControls.Button
32. Protected WithEvents SubmitButton As
System.Web.UI.WebControls.Button
33. Protected WithEvents RequiredFieldValidator1 As
System.Web.UI.WebControls.RequiredFieldValidator
34. Protected WithEvents RequiredFieldValidator2 As
System.Web.UI.WebControls.RequiredFieldValidator
35. Protected WithEvents RequiredFieldValidator3 As
System.Web.UI.WebControls.RequiredFieldValidator
36. Protected WithEvents RequiredFieldValidator4 As
System.Web.UI.WebControls.RequiredFieldValidator
37. Protected WithEvents RequiredFieldValidator5 As
System.Web.UI.WebControls.RequiredFieldValidator
38. Protected WithEvents RegularExpressionValidator1
As
System.Web.UI.WebControls.RegularExpressionValidator
39. Protected WithEvents cmdInsert As
System.Data.SqlClient.SqlCommand
40. Protected WithEvents con As
System.Data.SqlClient.SqlConnection
41. Protected WithEvents EMailTextBox As
System.Web.UI.WebControls.TextBox

42. 'NOTE: The following placeholder declaration is
required by the Web Form Designer.
43. 'Do not delete or move it.
44. Private designerPlaceholderDeclaration As
System.Object
```

```
45. Private Sub Page_Init(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Init
46. 'CODEGEN: This method call is required by the Web
Form Designer
47. 'Do not modify it using the code editor.
48. InitializeComponent()
49. End Sub

50. #End Region

51. Private Sub Page_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
52. 'Put user code to initialize the page here
53. End Sub

54. Private Sub SubmitButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SubmitButton.Click
55. Try
56. con.Open()
57. cmdInsert.CommandText = "Insert into Registeration
(Name,Password,BirthDate,Tel,EMail,Address) values ('"
+ NameTextBox.Text + "','" + PasswordTextBox.Text +
"','" + BirthDateTextBox.Text + "','" +
TelephoneTextBox.Text + "','" + EMailTextBox.Text +
"','" + AddressTextBox.Text + "')"
58. cmdInser..ExecuteNonQuery()
59. Response.Redirect("Thnax.ASPX")
60. Catch ex As Exception
```

```
61. statusLabel.Text = "Error in DataBase :" +
ex.Message
62. Finally
63. con.Close()
64. End Try
65. End Sub

66. Private Sub ResetButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ResetButton.Click
67. NameTextBox.Text = ""
68. PasswordTextBox.Text = ""
69. BirthDateTextBox.Text = ""
70. EMailTextBox.Text = ""
71. AddressTextBox.Text = ""
72. TelephoneTextBox.Text = ""
73. End Sub
74. End Class
```

**شرح الكود:**

- في السطور من 55 إلى 64 يتم معالجة الزر Submit عند الضغط عليه وهى كالآتى.
- فى السطر 56 نقوم بفتح الوصلة مع قاعدة البيانات وذلك باستخدام الدالة .con.Open()
- فى السطر 57 نقوم بتحديد أمر الـ SQL الذى سيؤديه الهدف cmdInsert ، وهذا الهدف object سيقوم بإدخال Insert بيانات جديدة الى قاعدة البيانات وذلك باستخدام أمر Insert وذلك كما فى السطر 89:

cmdInsert.CommandText="Insert into Registration (Name,Password,BirthDate,Tel,EMail,Address) values ('"+ NameTextBox.Text+"','"+PasswordTextBox.Text+"','"+BirthD

ateTextBox.Text+"','"+TelephoneTextBox.Text+"','"+EMailTextBox.Text+"','"+AddressTextBox.Text+"')";

حيث يتم أخذ قيم الحقول من أدوات النصوص TextBox الموجودة على النموذج Form.

- فى السطر58 يتم تنفيذ الأمر فى قاعدة البيانات عن طريق استدعاء الدالة Method:

```
cmdInsert.ExecuteNonQuery();
```

ويتم استدعاء الدالة ()ExecuteNonQuery عندما نقوم بعمليات تأثير فعلية فى البيانات مثل التعديل والإدخال والمسح (Update,Insert,Delete).

- فى السطر 59 يتم التحويل إلى الصفحة thnx.aspx إذا تم إدخال البيانات بدون أى أخطاء باستخدام:

```
Response.Redirect("Thnax.ASPX");
```

والهدف Response هو من الأهداف objects المدمجة (التى لا تحتاج إلى إنشائها قبل الاستعمال).

- فى السطور من 67 إلى 72 يتم معالجة حدث الضغط على الزر Reset وهى كما ترى بسيطة فكل ما نفعله هو أن نقوم بحذف القيم التى قد تكون فى أدوات النصوص TextBox الموجودة على النموذج Form.

وبذلك نكون قد انتهينا من هذا النموذج Form.

## العمل مع النموذج Thnx.aspx:

قم بالذهاب إلى نافذة الـ Solution Explorer وقم بالضغط مرتين Double-click على الصفحة Registration.aspx لتصبح فى حالة التصميم ثم قم بإضافة الأدوات التالية مع تغيير خصائصها كما هو واضح فى الجدول التالى ولتصبح كما هو واضح فى شكل 19 السابق.

| الأداة | الخاصية | قيمة الخاصية |
|---|---|---|
| Label 1 | Text | Thnx for Registration please Click this Button To View Our Titles |
| | Width | px425 |
| | Height | px24 |
| Button 1 | Text | View Book Titles |
| | Width | px24 |
| | ID | ViewBooksButton |

هذا النموذج Form بسيط كما ترى ، فقم بالضغط مرتين Double-click على الزر View Books Titles ، ثم قم بجعل الدالة Method تبدو كما فى الكــود التالى:

وكما هو واضح من الكود أنه عند الضغط علي الزر فيتم التحويل إلى النمــوذج BookTitles.aspx.

```
Private Sub ViewBooksButton_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ViewBooksButton.Click
    Response.Redirect("BookTitles.aspx")
End sub
```

وبذلك نكون قد انتهينا من هذا النموذج Form.

## العمل مع النموذج BookTitles.aspx:

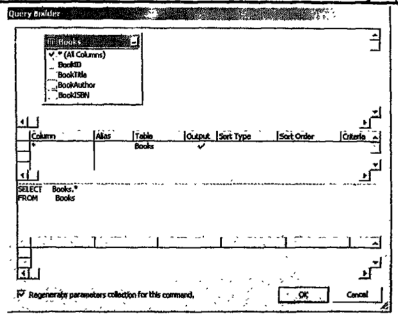
قم بالذهاب إلى نافذة الـ Solution Explorer وقم بالضغط مرتين Double-click على الصفحة Registration.aspx لتصبح فى حالة التصميم وقم بإضافة الأدوات التالية مع تغيير خصائصها كما هو واضح فى الجدول التالى ولتصــبح كما هو واضح فى شكل 20.

| الأداة | الخاصية | قيمة الخاصية |
|---|---|---|
| DataGrid | Width | px728 |
| | Height | px488 |

## العمل مع أدوات قواعد البيانات:

فى هذا النموذج Form سوف نتعامل مع قواعد البيانات ولذلك لابد من استخدام أدوات قواعد البيانات الموجودة فى الفئة Data الموجودة فى صندوق الأدوات ToolBox ، لذلك قم بسحب الأداتين SqlConnection, SqlCommand ووضعهما على النموذج Form وستجدهما ممثلتين بأيقونتين icons أسفل النموذج Form.

- قم باختيار الأداة SqlConnection ثم قم بضغط الزر F4 وذلك لضبط خصائصها وذلك كالآتى:
- قم بتغيير خاصية الاسم إلى con.
- قم بالذهاب إلى الخاصية connection String ثم اختر من القائمة المنسدلة اسم الخادم Server المسمى BookShow.dbo.
- قم بعد ذلك باختيار الأداة SqlCommand1 الموجودة أسفل النموذج Form ثم قم بضغط الزر F4 لضبط خصائصها.
- قم بتغيير خاصية الـ Name إلى cmdSelect.
- قم بالذهاب إلى خاصية connection ثم قم بالضغط على القائمة المنسدلة ثم قم بالضغط على علامة + الموجودة بجانب existing ثم قم باختيار con وهى الوصلة التى أعددناها فى الخطوة السابقة.
- قم بالذهاب إلى الخاصية CommandText ثم قم بالضغط على الزر المنقط فيظهر لك مربع حوار Dialog Box طالباً منك تحديد الجدول الذى ترغب فى العمل معه فاختر الجدول Books ثم اضغط الزر Add ثم اختر جميع الحقول وذلك كما هو واضح فى الشكل 35.




شكل 35 العمل مع قواعد البيانات

وبذلك نكون قد انتهينا من عمل الأدوات التي سنستخدمها فى العمل مع قواعد البيانات.

والكود التالى هو كود النموذج Form وهو يماثل ما هو موجود لديك فيما عدا المظلل منه فقم بإضافته وهو ما سنقوم بشرحه.

```
1.   Imports System.Data
2.   Imports System.Data.SqlClient
3.   Public Class BookTitles
4.   Inherits System.Web.UI.Page
5.   Private dtaAdapter As SqlDataAdapter
6.   Private ds As DataSet

7.   #Region " Web Form Designer Generated Code "

8.   'This call is required by the Web Form Designer.
9.   <System.Diagnostics.DebuggerStepThrough()>
Private Sub InitializeComponent()
```

```
10. Me.cmdSelect = New
System.Data.SqlClient.SqlCommand
11. Me.con = New System.Data.SqlClient.SqlConnection
12. '
13. 'cmdSelect
14. '
15. Me.cmdSelect.CommandText = "SELECT Books.*
FROM Books"
16. Me.cmdSelect.Connection = Me.con
17. '
18. 'con
19. '
20. Me.con.ConnectionString = "workstation
id=ISIS;packet size=4096;integrated security=SSPI;data
source=isis;pe" & _
21. "rsist security info=False;initial catalog=BookShow"

22. End Sub
23. Protected WithEvents DataGrid1 As
System.Web.UI.WebControls.DataGrid
24. Protected WithEvents cmdSelect As
System.Data.SqlClient.SqlCommand
25. Protected WithEvents con As
System.Data.SqlClient.SqlConnection

26. 'NOTE: The following placeholder declaration is
required by the Web Form Designer.
27. 'Do not delete or move it.
28. Private designerPlaceholderDeclaration As
System.Object
```

```
29. Private Sub Page_Init(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Init
30. 'CODEGEN: This method call is required by the Web
Form Designer
31. 'Do not modify it using the code editor.
32. InitializeComponent()
33. End Sub

34. #End Region

35. Private Sub Page_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
36. InitializeDB()
37. End Sub
38. Sub InitializeDB()
39. Try
40. con.Open()
41. dtaAdapter = New SqlDataAdapter(cmdSelect)
42. ds = New DataSet
43. dtaAdapter.Fill(ds, "BookTitle")
44. con.Close()
45. DataGrid1.DataSource = ds
46. DataGrid1.DataMember = "BookTitle"
47. DataGrid1.DataBind()
48. Catch ex As Exception
49. Response.Write(ex.Message)
50. Finally
51. con.Close()
52. End Try
53. End Sub
54. End Class
```

## شرح الكود:

- في السطرين 1 و 2 قمنا باستيراد المكتبات namespace التي تقع فيها الفصيلة SqlDataAdapter والفصيلة DataSet.
- في السطر 5 قمنا بالإعلان عن مؤشر dtaAdapter من نوع SqlDataAdapter وهو -كما ذكرنا سابقاً- فصيلة class بمثابة الشاحنة التي يتم ملؤها بالبيانات من قاعدة البيانات ثم يتم تفريغها في الـ DataSet.
- في السطر 6 قمنا بالإعلان عن مؤشر ds من نوع DataSet.
- في السطور من 38 حتى 53 قمنا بإنشاء الدالة IntializeDb() وهي دالة Method مهمتها القيام بالاتصال بقاعدة البيانات وملئ أداة شبكة البيانات DataGrid فتعال معي نتابع سطور هذه الدالة Method:
- في السطر 40 نقوم بفتح الوصلة عن طريق الدالة con.Open().
- في السطر 41 قمنا بإنشاء هدف object للمؤشر dtaAdptr وذلك عن طريق الجملة:

```
dtaAdptr=new SqlDataAdapter(cmdSelect);
```

وقد أعطينا دالة البناء constructor معاملاً parameter هو اسم الهدف cmdSelect الذي يتم تنفيذ أوامره (select * from books) فيتم ملئ الهدف dtaAdptr بالبيانات.

- في السطر 42 نقوم بإنشاء هدف object للمؤشر ds وذلك عن طريق الجملة:

```
ds=new DataSet();
```

- في السطر 43 يتم ملئ الهدف ds بالبيانات الموجودة في الهدف dtaAdptr وذلك عن طريق الدالة Fill() والتي أعطيناها معاملين Parameters هما اسم الهدف ds والثاني الاسم الذي سنعطيه لهذا الجدول وهو BooKTitle

```
dtaAdptr.Fill(ds,"BooKTitle");
```

- في السطر 44 نقوم بإغلاق الوصلة.
- في السطر 45 نقوم بتحديد مصدر البيانات الذى سيتم ملئ الأداة DataGrid منه وطبعاً هو الهدف ds.
- في السطر 46 نقوم بتحديد اسم الجدول الذى تكون محتوياته هو البيانات فى الأداة DataGrid.
- في السطر 47 نقوم بربط الأداة DataGrid وذلك لإظهار البيانات بها.
- في السطر 36 قمنا باستدعاء الدالة ;()IntializeDb في الدالة ()Page_Load وذلك ليتم تنفيذها عند تحميل Load الصفحة.

وبذلك نكون قد انتهينا من شرح جميع النماذج Forms في هذا المثال ويمكنك الآن تنفيذ التطبيق والتأكد من عمله بالشكل الصحيح وسنترك هذه الخطوة كتمرين لك.

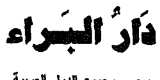
# فهرس المحتويات

# من إصدارات دار البراء

| اسم الكتاب | المؤلف |
|---|---|
| تعلم بدون تعقيد .. تجميع وصيانة الكمبيوتر | م/أحمد حسن خميس |
| استخدام البالتوك | محمد نزيه محمد |
| تعلم بدون تعقيد .. تثبيت وصيانة ويندوز | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. برامج المنافع (Norton Utilities) | أ/ شريف محمد سعيد |
| تعلم بدون تعقيد .. استخدام الإنترنت | م/أحمد حسن خميس |
| تعلم بلا حدود .. فيجوال بيسك 6.00 ( المهام الأساسية ) | أ/ سامي عبد العزيز |
| تعلم بلا حدود .. فيجوال بيسك 6.00 ( المهام المتقدمة ) | أ/ سامي عبد العزيز |
| تعلم بلا حدود .. فيجوال بيسك 6.00 ( قواعد البيانات ومهام الإحتراف ) | أ/ سامي عبد العزيز |
| تعلم بدون تعقيد .. ويندوز 98 | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. استخدام ويندوز إكس بي | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. استخدام وورد إكس بي | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. استخدام إكسيل إكس بي | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. استخدام أكسيس إكس بي | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. استخدام بوربوينت إكس بي | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. أدوب فوتوشوب 7.00 | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. فلاش إم إكس | م/مصطفى ماجد |
| تعلم بدون تعقيد .. أوتوكاد 2004 | م/أحمد حسن خميس |
| تعلم بدون تعقيد .. ثري دي ماكس 5.00 | م/محمد العوضي |
| تعلم بدون تعقيد .. الشبكات ( التصميم والتركيب والصيانة) | أ/ شريف محمد سعيد |
| الهاكرز والكراكرز (إختراق الأجهزة الشخصية) | م/أحمد حسن خميس |
| دليل مواقع الإنترنت | م/أحمد حسن خميس |
| احترف .. إكسيل إكس بي ( الدوال و الماكرو والبرمجة ) | م/أحمد حسن خميس |
| احترف .. أكسيس إكس بي ( البرمجة والمهام المتقدمة ) | م/أحمد حسن خميس |
| احترف .. HTML (تكنولوجيا CSS) | م/مصطفى ماجد |
| احترف .. JAVA Script | م/مصطفى ماجد |
| احترف .. فلاش إم إكس (أكشن سكريبت والمهام المتقدمة) | م/مصطفى ماجد |

| اسم الكتاب | المؤلف |
|---|---|
| من البداية إلى الإحتراف .. التركيب والصيانة (الدش 1) | أ/ مدحت أبو الحسن |
| من البداية إلى الاحتراف .. الأعطال والصيانة (الدش2) | أ/ مدحت أبو الحسن |
| من البداية إلى الاحتراف .. الشبكات المركزية (الدش3) | أ/ مدحت أبو الحسن |
| من البداية إلى الإحتراف .. صيانة الموبايل | أ/ محمد عيد المصرى |
| تعلم بلا حدود .. برمجة ألعاب الكمبيوتر | م/أحمد حسن خميس |
| تعلم بلا حدود .. ويندوز إكس بى | م/أحمد حسن خميس |
| تعلم بلاحدود .. أوراكل 9i (SQL) | م/حسام الدين إمام |
| تعلم بدون تعقيد .. DOS (عندما تفقد السيطرة على ويندوز) | أ/ شريف محمد سعيد |
| تعلم بدون تعقيد .. تطبيقات فوتوشوب | ا/ عبد المنعم فريد |
| أسرار وأفكار ويندوز إكس بى | أ/ كمال علام |
| رسائل الخطأ (معناها ..أسبابها .. علاجها) | أ/ شريف محمد سعيد |
| تعلم بدون تعقيد .. Dreaweaver MX | م/مصطفى ماجد |

والعديد من الإصدارات الأخرى




بمصر وجميع الدول العربية

# طبعة يناير 2005

رقم الإيداع

2005/1912

**ISBN**

977-17-1971-8




المركز الرئيسى : 11 شارع د/محمد رأفت – محطة الرمل – الإسكندرية

تليفون وفاكس : 4838326 (03)(+2)

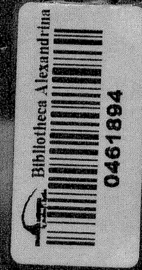
موبايل : 0101634294 (+2) - 0123357844 (+2)

**Email :** info@egypt*books*.net

**URL:** www.egypt*books*.net




- Introduction to .NET مقدمة إلى دوت نت
- Data Types أنواع البيانات
- Control Statements جمل التحكم
- Arrays and Methods المصفوفات والدوال
- Classes and Objects الفصائل والأهداف
- Inheritance, Sealed Classes And Modifiers الوراثة والفصائل النهائية والمعدلات
- Polymorphism and Abstraction تعدد الأشكال والتجريد
- Exceptions الاستثناءات
- Building Graphical User Interface(GUI) بناء واجهة المستخدم الرسومية
- Event Handling معالجة الأحداث
- ActiveX Data Object(ADO.NET) هدف البيانات
- Active Server Pages(ASP.NET) صفحات الخادم النشطة








للدعم الفنى تفضل بشبكة الكتب المصرية

www.egyptbooks.net

الموقع الرسمى لدار البراء



المركز الرئيسي : ١١ شارع د/محمد رأفت – محطة الرمل – الإسكندرية

تليفون وفاكس : 4838326 (03)(+2)

موبايل : 0101634294 (+2) - 0123357844 (+2)